Bundles in Captivity: An Application of Superimposed Information[1]

Lois Delcambre[†], David Maier[†], Shawn Bowers[†], Mathew Weaver[†], Longxing Deng[†], Paul Gorman[‡], Joan Ash[‡], Mary Lavelle[‡], and Jason A. Lyman[‡]

[†]*Oregon Graduate Institute*
{lmd, maier, shawn, mweaver, longxing}@cse.ogi.edu

[‡]*Oregon Health Sciences University*
{gormanp, ash, lavellem, lymanja}@ohsu.edu

Abstract

What do you do to make sense of a mass of information on a given topic? Paradoxically, you likely add yet more information to the pile: annotations, underlining, bookmarks, cross-references. We want to build digital information systems for managing such added or superimposed *information and support applications that create and manipulate it.*

We find that requirements for a superimposed information system can be quite different from those for a traditional database management system: a lightweight implementation, multi-model information structures, "schema-later" data entry, interacting with data that is "outside the box" (controlled by other applications), and support, rather than removal, of redundancy.

We report here on SLIMPad, a superimposed application, which was inspired by the "bundling" of information elements from disparate sources we observed in a medical setting. We propose an architecture for superimposed applications and information management. Our prototype components to implement the architecture give flexibility in structuring superimposed information, and also encapsulate addressing, at a sub-document granularity, into a variety of base information sources.

1. Superimposed information and superimposed applications: Our vision

Consider a concordance for the works of Shakespeare. For a given term, we can find out every line (in a play) where the term is used. A concordance is one example of what we call *superimposed information*, where supplemental information is created to reference, highlight, and extend information present elsewhere. Other examples of superimposed information are citation indices and commentaries. Superimposed information relies on an addressing scheme for information elements in the original documents, often at a fine granularity, e.g., play-act-scene-line. Typically we find such addressing schemes in place only for important or widely used information sources in the traditional print literature.

The ever-increasing number of electronic information sources accessible to a worldwide audience provides an unprecedented opportunity to consider generic mechanisms for supporting superimposed applications. Such applications allow one to create and exploit superimposed information, in digital form, with those sources as a base layer. Naturally there are associated challenges – the wide range of information types, the many base-layer applications that manage information, accommodating variety of information structures and models in the superimposed layer, and building generic technology that can be deployed in various superimposed applications.

Our view of superimposed information [6, 16] is shown in Figure 1. The superimposed layer is conceptually and possibly physically distinct from the base information sources, and may conform to a data model or other information structure. The superimposed information may contain *marks* to selected information elements in the base layer, where a mark holds a suitable address for an element. The base layer may include multiple, heterogeneous sources. We require only that the base layer support a local addressing scheme for information that it contains and that it permits creation and resolution of marks.

Many superimposed applications already exist, e.g., that provide for annotation [23], shared bookmarks [14], and constructing virtual documents [11]. And we see models for information emerging that are inherently superimposed including topic maps [3], RDF [12], and XLink [7]. Our vision is a generic technology to manage

[1] This work is supported, in part, by the National Science Foundation, Grant Number II-98-17492 as part of the Digital Libraries Initiative II, funded by NSF, DARPA, NLM, Library of Congress, NEH and NASA.

superimposed information and support the construction of superimposed applications.

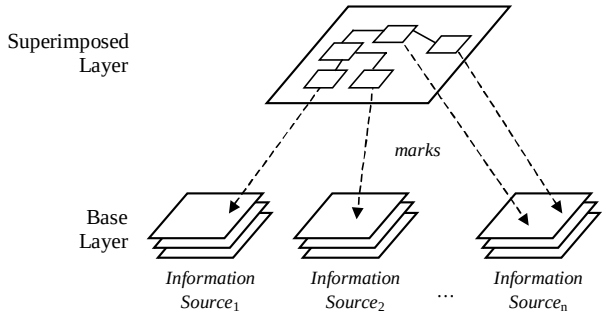

Figure 1. Superimposed information with marks referencing the base-layer.

We propose an architecture for superimposed information management, and a prototype application, SLIMPad, built according to the architecture. SLIMPad (Superimposed Layer Information Manager scratchPad) is inspired by observational work in a hospital intensive care unit on how clinicians select and use information. It supports freeform collections of information elements selected from multiple base sources, with linkages back into the original contexts of those elements. While the superimposed information model is quite simple, the representation, storage and management of superimposed information inside SLIMPad is generic and able to accommodate a range of information structures.

In designing our architecture, and in constructing initial implementation of its components, we adhered to three principles.

Keep it lightweight: In most of the applications we've studied or contemplated, the superimposed information is a thin layer over more extensive information sources in the base layer. The application using the superimposed information might exist as an extension or plug-in to an existing application. Requiring a heavy-duty data manager, with a large code footprint, to manage the superimposed information is undesirable.

Keep it flexible: We have observed a wide range in data models for superimposed information. Thus we want an information management capability that isn't wired to a single data model. SLIM thus contains data-model-definition capability, in addition to the normal schema-definition capability of a data manager.

Minimize assumptions about the base layer: At least initially, we want to work with as many different kinds of base layer sources as possible. Accordingly, we make few assumptions about the capabilities of the managers of the base sources. In particular, we do *not* assume they are full-fledged database systems. Rather, we assume only that a base source can supply the address of a currently selected information element, and that it can return to that element given the address. While these capabilities may seem hopelessly limited, we have built a useful application on top of them. Further, this very narrow interface to the base layer has made our architecture readily extensible and has hidden variations in base layer managers from the rest of the system, greatly aiding concurrent development of our technology.

2. Bundles in the wild: What we've observed

In our field observations of expert clinicians caring for patients in critical care units, bundles appear to be a widely used means of managing information to support diverse, complex, often simultaneous tasks [8]. We define a *bundle* as a grouping of information selected, collected, elaborated, and structured by a clinician during problem solving. A bundle might be jottings on the back of an envelope, items organized on a blank sheet of paper with headings and groupings, or entries on a printed card.

Regardless of the media, bundles are freeform; both the content and the structure are created by and for the clinician. Figure 2 shows several examples of *bundles in the wild*, i.e., bundles that we have observed in our work. At top, we see two bundles that have been written on available scraps of paper (an unopened gauze pad and paper towel). On the upper left we see (underneath the pad) a more structured bundle called a *flowsheet*, where the status of an intensive-care patient is tracked over time. Bundles often contain (selected) information that is available elsewhere. Bundles may also capture information that will be recorded elsewhere later. The bottom of Figure 2 shows one row (corresponding to one patient) of a resident's worksheet, prepared in advance of making patient rounds, and updated during rounds. The first column identifies the patient, the second lists significant problems, the third contains selected lab results and vital signs, and the last is a to-do list. The multiple rows on the worksheet illustrate another observation: bundles can be grouped into larger bundles.

Bundles may be especially useful in settings, such as intensive care, with high uncertainty, low predictability and potentially grave outcomes, where time and attention are constrained, and where interdisciplinary teamwork is essential. Reports of observations from other, analogous domains such as air traffic control suggest that bundle use may be common outside the medical area [9, 10, 15].

We believe there is benefit in creating bundles (through the active processing of information to improve understanding), in reusing bundles (by triggering memory) and in sharing bundles to establish collectively maintained, situated awareness. The selection of bundle content itself adds value by excluding information that's not considered important to the current context.

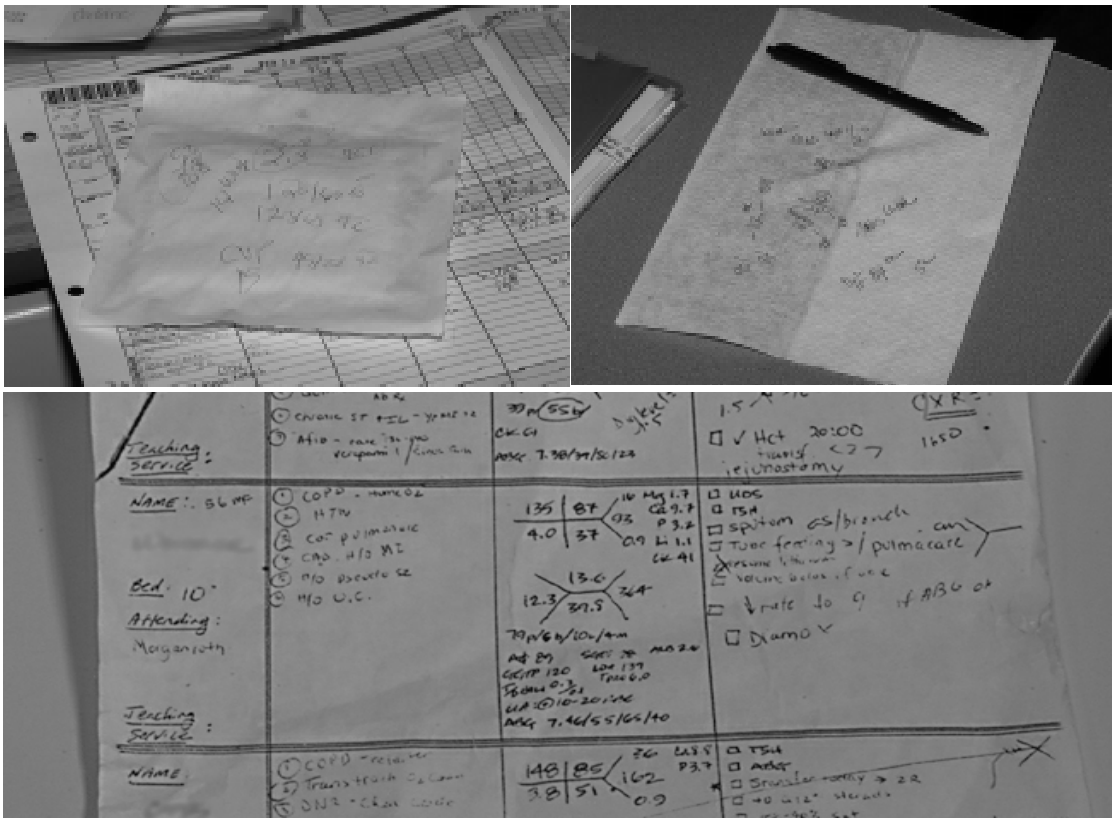

Figure 2. Selected bundles from the Intensive Care Unit.

3. SLIMPad: What we've built

To test our superimposed information architecture, we built a superimposed application to represent bundles digitally. While we didn't expect to completely duplicate the flexibility of paper, we hoped to gain offsetting advantages from the digital format. As noted, bundles in the wild often contain information present in other sources. While redundancy in information systems may be undesirable in general, redundancy in bundles can be useful. The selection and rearrangement of information in a bundle can reveal patterns not readily apparent with the information in situ. A bundle can also serve as a quick reminder of the most significant pieces of information in a given situation. Redundancy is a problem, however, if it introduces errors during transcription. Thus we decided to link information elements that come from digital sources to their location in those sources, to minimize inconsistency. Using these links, we can re-establish context for a selected item, and navigate to nearby information.

Based on our field observations, we decided that bundles should be constructed manually rather than by using rules or views. We had observed that bundle content and organization is highly variable across people and situations. A particular bit of information might be essential in one patient's case yet irrelevant for another. Also, manual construction involves active processing of information, thus generates awareness of it, and provides evidence to others of that awareness.

For the information model and user interface to digital bundles, we wanted simple models, to make the application flexible, easily understood, and unlikely to exclude a piece of information the user wants to add. We do not prescribe any particular content in a digital bundle. And we permit users to add information elements without prior definition of their meaning or their grouping into bundles. That is, we present more the "information-first" flavor of a word processor or spreadsheet, rather than the "schema-first" approach of a database. We allow flexibility for placement of information elements and bundles in two dimensions. The juxtaposition of scraps and bundles contains implicit semantic information that we neither want to constrain or lose.

The SLIM scratchPad (SLIMPad) allows users to create structured, digital, bundles. Although SLIMPad is not intended to support any specific, observed medical task, it has allowed us to explore technology for superimposed information in parallel with the observational work. Many bundles in the wild have a scratchpad-like feel and appearance. SLIMPad provides this same look and feel, in a

computerized tool. SLIMPad's information model is shown in Figure 3, represented in UML [22].

The model consists of four main entities. The top-level object is a *SLIMPad*, which designates a root bundle. Each *Bundle* has a label and position, and can contain any number of Scraps or Bundles. A *Scrap* (i.e., information element) has a label and a MarkHandle object. A *MarkHandle* has a mark identifier, which refers to a Mark object inside the Mark Manager. The Mark Manager is a component of our architecture responsible for interacting with base applications. The bottom of Figure 3 shows the information model of the MarkManager. A *Mark* contains the address to the marked information element, in whatever form required by the base source. There is one subclass of Mark for each type of base information supported.

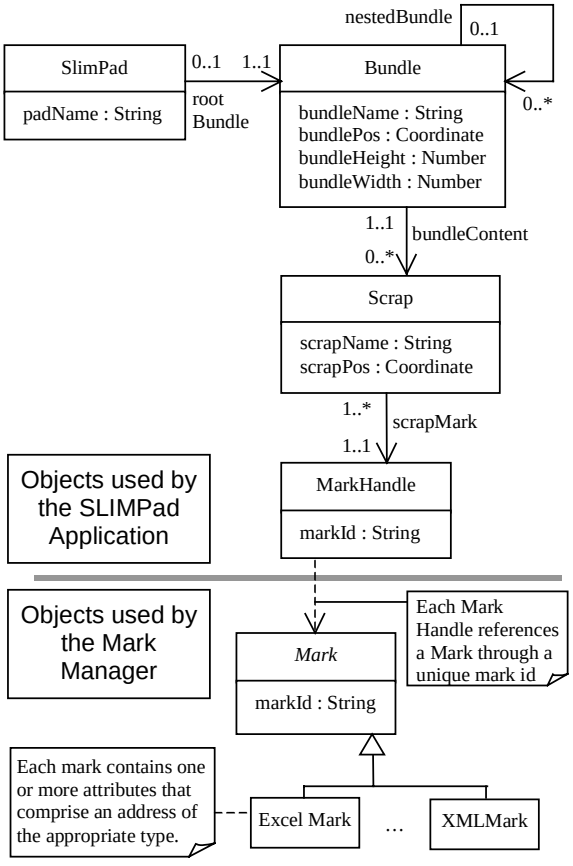

Figure 3. The Bundle-Scrap data model.

The SLIMPad application provides an interface between the user and data in this model. Each visual entity the user sees on the screen corresponds to an object in the data model. Figure 4 shows the SLIMPad application being used on a medical example, inspired by the resident's worksheet in Figure 2. The largest window, titled 'Rounds', is the visual representation of a SLIMPad object. In this case, the user has created a bundle, titled 'John Smith'. The bundle contains three scraps and another bundle. The top two scraps represent medications for the patient. The mark associated with each scrap refers to the corresponding medication in a complete medication list (here, a Microsoft Excel document). By clicking on the scrap, the mark is de-referenced and the original information source, the medication list, is displayed with the appropriate medication highlighted, as shown in the upper right of Figure 4. Note that a scrap's label and its mark's content may differ.

The 'Electrolyte' bundle contains a set of scraps that come from a lab report, represented in an XML document. Each of these scraps can be double-clicked, which opens the lab report and highlights the appropriate section of the XML document. The "gridlet" in this bundle is simply a graphic element with scraps placed near it.

To create a new scrap, the user selects an information element using a base-layer application (such as Excel) that has been modified to support the creation of marks. Once the user has created a mark, it can be placed onto the SLIMPad, creating a scrap that can be named and moved around. By creating the mark and attaching it to a scrap, the user creates a digital "sticky-note," which comes with a digital "wire" that leads back to the information in the original data source. Note that it is the user who determines the physical layout of the scraps and the bundles. For example, each number in the 'Electrolyte' bundle has a specific meaning to a medical professional, which can be deduced from their arrangement relative to each other. The SLIMPad data model does not impose structure – but allows the user to create structure.

SLIMPad currently supports marks into Microsoft Excel® spreadsheets, Word® documents, and PowerPoint® presentations; XML documents; Adobe® PDF documents; and HTML pages. The SLIMPad application is our first that supports digital bundles, with an intentionally simple model. We are also investigating digital bundles in a more general context where the model may be more complex, e.g., with more structure, with multiple marks per scrap, with explicit links between scraps.

4. Generic technology for superimposed information: How we build it

Figure 5 shows our architecture for managing superimposed information. A superimposed application (e.g., SLIMPad) interacts with base-layer applications and the generic management components. The generic components serve two main purposes: manage marks to base-layer data and manage the application's superimposed information (e.g., bundles and scraps).

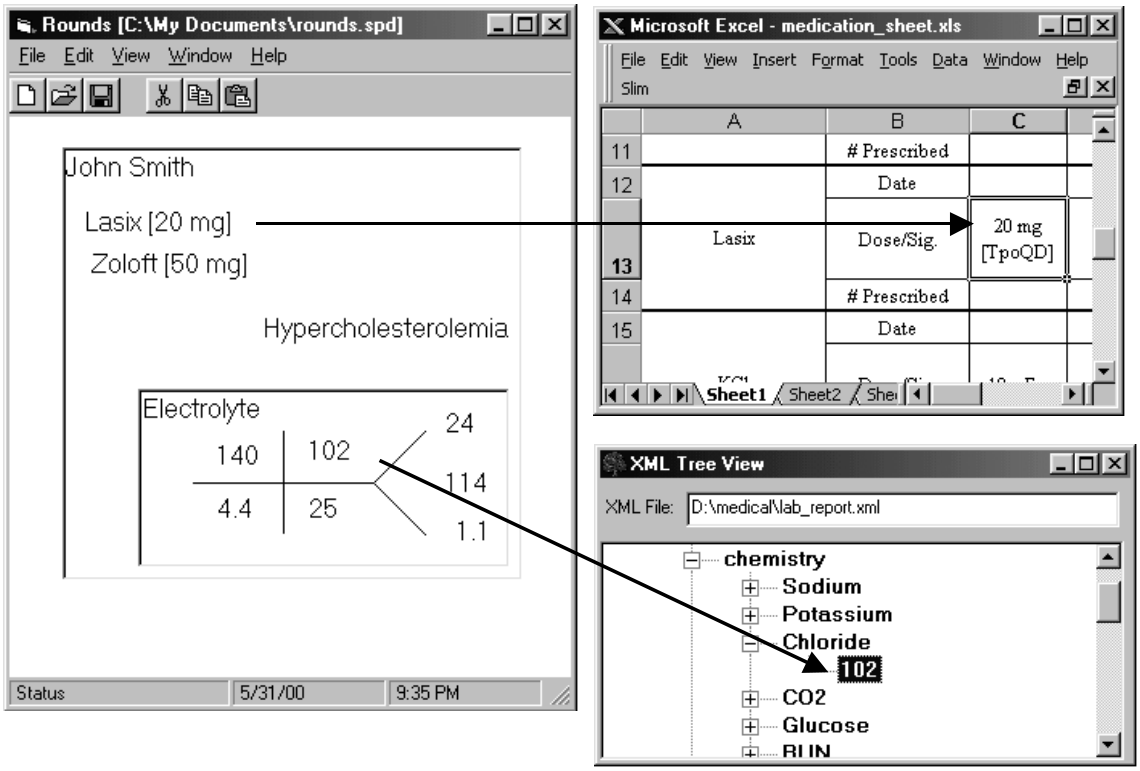

Figure 4. SLIMPad screenshot showing two marks: one to Excel and one to XML.

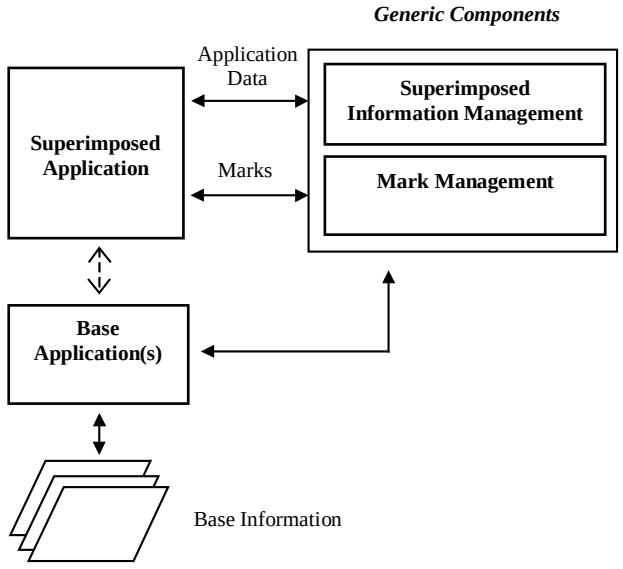

Figure 5. Overview of our superimposed application architecture.

4.1. Base Application

A base application can be any one that acts as an interface to base-layer information, such as a word processor, a web browser or a database system. A superimposed application can interact with one or more base applications using three basic approaches. These approaches are defined by how the user views and interacts with the superimposed application and the base application (see Figure 6).

With *simultaneous viewing*, a user accesses superimposed and base-layer applications at the same time. Usually, there are two windows active on the computer screen: one for the superimposed application and one for the base application. A user interacts with either window as desired. With *enhanced base-layer viewing*, the functionality of a base application is enhanced to manage superimposed information. Third Voice [23] is such an example, which enhances web browsers by allowing the user to create and view annotations in the same browser window as the Web page. With *independent viewing*, the base application is hidden. A user sees only the superimposed application, which may expose the functionality of the base application, usually in a limited way. The base application can work in the background to extract base information elements for the superimposed layer, or it can work as an in-place viewer for base information.

SLIMPad is designed to support simultaneous viewing,. In normal operation, the user juxtaposes the SLIMPad window and base document viewer(s) to interact with base and superimposed information simultaneously, as shown in Figure 4. SLIMPad can support independent

viewing by having marks on the SLIMPad resolve to display the content of the marked element in place.

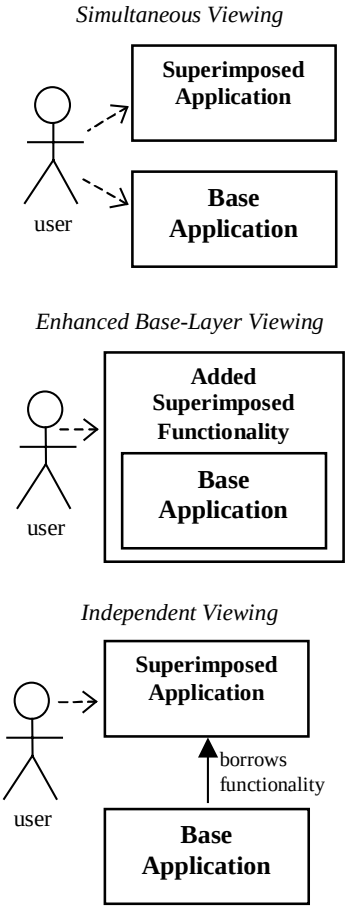

Figure 6. Three viewing styles for superimposed applications.

4.2. Mark Management

A fundamental objective of digital superimposed information is maintaining a link to the base-layer information. The *Mark Manager* is the framework for creating and managing these links – called *marks*. A *mark module* works with each base-layer application to create and resolve marks. Figure 7 shows the general approach of mark management (within the SLIMPad implementation).

A mark is stored and maintained in the superimposed information layer, but references information in the base layer. The information contained in a mark includes an address specific to the base-layer information. Each type of base-layer information has its own type of mark, represented as a subclass of Mark, as shown in Figure 3. Figure 8 shows the internal structure of two mark types.

A Microsoft Excel mark addresses information stored in an Excel workbook. This type of mark refers to a cell or range of cells within the workbook, using row and column positions. An XML mark references an element within an XML file. A new type of base-layer information is added by creating a new mark type. Since the specific addressing scheme of the base-layer information is encapsulated within the mark, the Mark Manager can generically store and retrieve all marks.

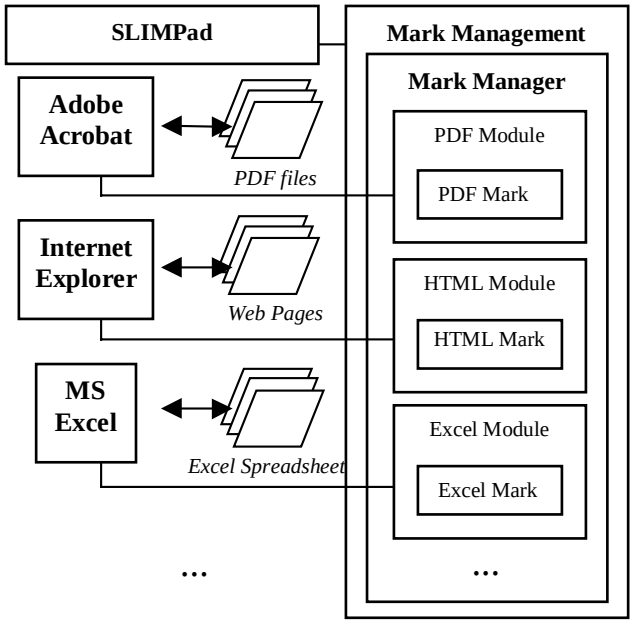

Figure 7. Architecture for mark management (as implemented in SLIMPad).

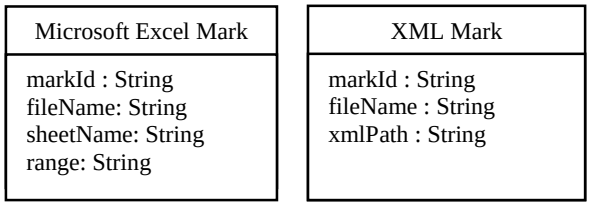

Figure 8. Excel mark and XML mark.

A mark is created by a base-layer application interacting with a *mark module*. A mark is specific to a certain type of base-layer information and a mark module is specific to a certain base-layer application. For example, an XML mark references XML data – but there may be multiple applications used to view the XML data. A mark module, specific to a base-layer application, enables the creation of marks by receiving information from that application and using it to create a mark. An Excel mark is created when Microsoft Excel gives the Excel mark module information containing the current selection within the current workbook. The Excel mark module then uses this information to create an Excel mark. A mark module resolves a mark by driving the base-layer application to the information element designated by the mark. The Excel mark module uses the address in an Excel mark object to

tell Microsoft Excel to open the file, activate the worksheet, and select the appropriate range.

To support new base-layer applications, new mark modules need to be introduced. This process includes enhancing the base-layer application to support the creation of marks and to react appropriately when asked to resolve a mark.

Mark management hides the details of the different kinds of base-layer information and base-layer applications from the superimposed application. From the superimposed application's viewpoint, a base information element is addressed by a mark, regardless of its type. This transparency simplifies the development of superimposed applications. The architecture is also easy to extend; new kinds of base information have been introduced without disturbing existing superimposed applications.

4.3. Superimposed information management: The SLIMStore

There are a number of existing superimposed models each of which differ in their structural characteristics (e.g., RDF, Topic Maps, XLink). Additionally, as superimposed applications are developed, the need for more specialized models will increase, SLIMPad's Bundle-Scrap model being one such example. Because of this variety, we choose to be flexible at the data-model level by providing storage of superimposed information for various models. Here we summarize our representation scheme for model-based superimposed information (described elsewhere [4]) and we say how the representation is used in SLIMPad.

The generic representation scheme for superimposed information is based on the metamodel, which can describe multiple superimposed-models. The metamodel consists of a basic set of abstractions to define model constructs and relationships (called connectors). For example, in the relational model, tables, attributes, keys and domains are constructs. The notion that tables contain attributes and attributes can be foreign or primary keys are implicit connections among the constructs defined by the model. Likewise, classes, attributes, and objects are constructs in an object-oriented model, in which there are implied connections between classes and attributes as well as classes and objects. The metamodel makes explicit the constructs of the model, their structural definitions, and their connections. Our goal is to create a metamodel that contains the universal set of modeling primitives used by models to define structure. Currently, the metamodel contains only a subset of primitives: constructs, which define a unit of structure; literal constructs for primitive type definitions; mark constructs for delineating marks; connectors, which describe basic relationships; conformance connectors for schema-instance relationships; and generalization connectors for specialization relationships.

We represent the metamodel elements using RDF Schema [5]. Superimposed model, schema, and instance data is represented using RDF triples (a triple is composed of a property, a resource, and a value).

There are a number of benefits to the generic representation. First, we can describe superimposed information from various models uniformly using RDF triples. Also, since RDF defines a serialization-syntax (in XML), we can use the representation for interoperability between superimposed applications. We can leverage the generic representation directly, by defining mappings between superimposed models, including model-to-model, schema-to-schema and even schema-to-model mappings [4].

4.4. The application-specific Data Manipulation Interface (DMI)

Although superimposed applications can use the generic representation directly to store and manipulate data, that would significantly complicate the development of a superimposed application. We describe an approach that lets an application manipulate data in its desired format, while storing the data using our generic representation (Figure 9). The superimposed application interacts with application data, which for SLIMPad are read-only objects that represent the Bundle-Scrap model of Figure 3, plus an application-specific Data Manipulation Interface (DMI). The DMI contains the allowable operations on the application's model. For example, SLIMPad can create, update, remove, store, and load objects (e.g., Bundles and Scraps) using its DMI.

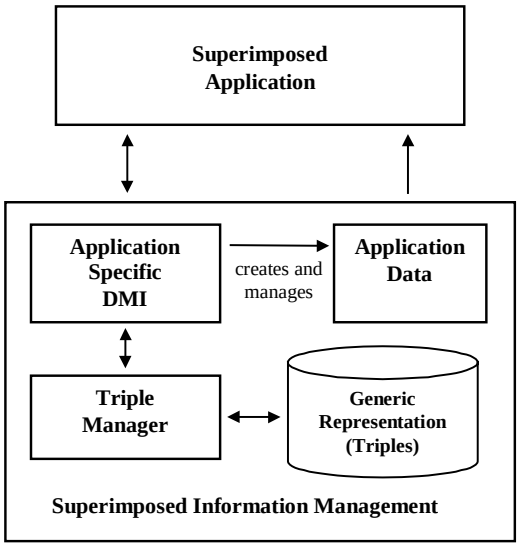

Figure 9. The superimposed layer information-management (SLIM) architecture.

When SLIMPad needs to create a Bundle, it calls the Create_Bundle operation in the DMI, which creates a Bundle object for SLIMPad plus the triples to represent a new Bundle. By restricting manipulation of data through the DMI, we store the triples without intervention from the superimposed application.

To manage triples, we use the TRIM (Triple Manager) sub-component, which handles basic operations over the triple representation. Through TRIM, the DMI can create, remove, persist (through XML files), query, and create simple views over the underlying triples. Query is specified by selection, where one or more of the triple fields is fixed, and the result is a set of triples. A view is specified by selecting a resource (such as a Bundle id), where all triples that can be reached from this resource are returned (e.g., all triples representing nested Bundles within the given Bundle along with their Scraps).

Figure 10 shows the internal data structures that are used by SLIMPad's DMI along with a portion of the DMI itself. The class structure is identical to the Bundle-Scrap model of SLIMPad, except the classes are writable (i.e., the DMI can set their attributes). For the SLIMPad application, we specfiy application data as a set of interfaces, where each construct in the model becomes a separate interface. The classes in Figure 10 implement each corresponding application data interface. Only the interfaces are presented to SLIMPad, which allows the DMI to guarantee consistency between the triple representation and the application data.

For SLIMPad, we generated the application data structures and DMI manually, based on the application model. We are working towards automatically generating specialized DMIs from data models (specified in either UML or as triples) [24].

5. Related Work

In a previous paper [6], several dimensions are used to compare different models of superimposed information. Here, we summarize the dimensions, use them to compare the Bundle-Scrap model to other superimposed models (from the earlier paper), and compare SLIMPad and our architecture to other approaches.

We describe superimposed information-space via three primary dimensions: the relationship between the base and superimposed layers; the addressability of marks; and the structural complexity of the superimposed model. The Bundle-Scrap model stands out in that it is open to various base layers and can address information granularities provided by base-layer applications. However, the model is not as complex as other models in terms of its ability to type marks, information elements, and links between information elements.

SLIMPad's capabilities differ from similar applications that support digital annotation. Annotation capabilities are typically part of a larger document editing or viewing system, such as Adobe Acrobat and Microsoft Word Comments. Complete systems also exist to provide shared-access to annotations over the Web [21, 23] and to annotate heterogeneous information sources [13, 20].

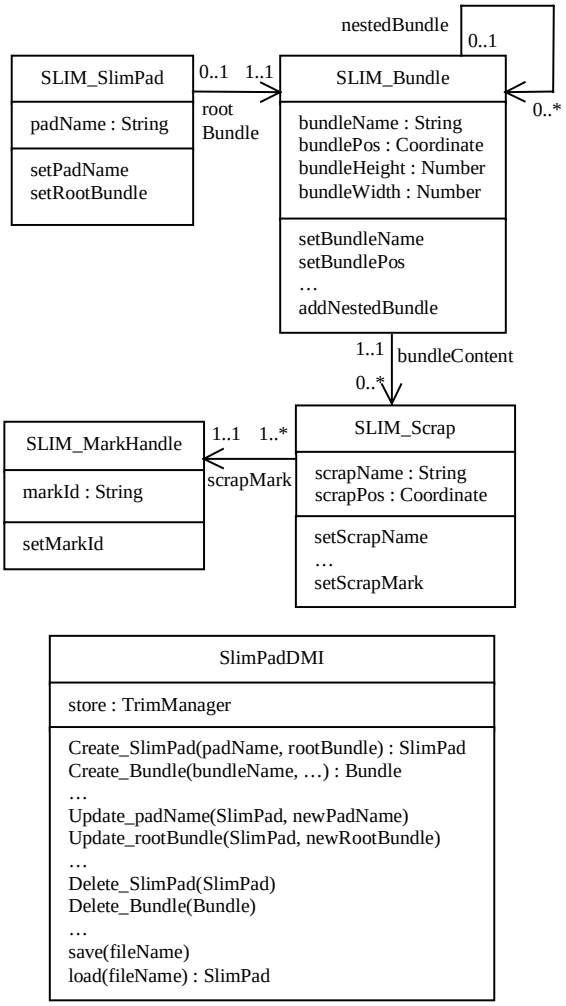

Figure 10. The objects used by the SLIMPad DMI.

In most annotation systems, users manipulate, create, and view annotations *in-situ* (annotations are available only while the document is being displayed). SLIMPad makes information available apart from the base documents themselves, but makes those documents available for context when requested. Additionally, SLIMPad's design is influenced by its use in very specific situations: by experts who wish to organize highly relevant information selections (e.g., to develop a medical hypothesis in an ICU). Bundles often contain excerpts or values from the original documents to provide emphasis and channel

awareness. (Some initial feedback from clinicians indicates annotations on scraps would be useful.)

Some systems store annotations separately from the documents they annotate. In ComMentor [21], users can ask for specific types of annotations created within a time range and use the returned annotations to navigate the corresponding web pages. SLIMPad is similar in that a separate structure can be used to access a base of underlying documents. However, SLIMPad's Bundle-Scrap model provides a richer abstraction than the annotation model of ComMentor. SLIMPad is also capable of marking a wider range of base layers than just HTML. Microsoft Comments, in which users can go to the next or previous annotation in a single document, and the Knowledge Weasel [13] also allow navigation through annotations.

Most annotation systems provide point and span marks for a specific place or a region in a document, respectively. Multivalent Documents (MVD) [20] use the structure of documents for addressing while accommodating a wide range of document types. This approach contrasts to annotation systems that use x and y coordinates within a viewer, or use document structure but for a single type of document (e.g., Web pages only). By using the inherent structure of documents, MVD can provide more complex types of marks such as *NoteMarks*, which combine several kinds of annotations together to serve as an index.

SLIMPad's approach for marking information sources is more generic than MVD. Instead of being document-centric, we choose to be application-centric, which means we can leverage the application's addressing mechanisms to provide various granularities. Our approach to marking documents and MVD are complementary in that we could incorporate MVD's viewer as a supported application.

Our mark architecture is similar to Microsoft Monikers [17]. Monikers allow components to obtain pointers to their referenced objects, which applications can then use directly. Both our architecture and Monikers provide application-interpreted addresses. That is, addresses must be resolved using the Moniker itself or a Mark Manager. The difference between our architecture and Monikers is that we use Mark Managers to resolve Marks instead of the Mark itself, which allows for multiple ways to resolve marks via different managers. For example, one manager for Excel can display Excel Marks in context and another act as an in-place viewer.

Composite document creation [11, 18] is an application type related to SLIMPad. Mirage-III [18] is a digital library system that allows users to create virtual documents (VDOCs) that contain span links to other documents. When a VDOC is rendered, the span links are resolved and the information they reference is displayed. The main difference between SLIMPad and virtual documents is that SLIMPad can contain information not present in the underlying documents.

Finally, our goal in managing superimposed information is to be flexible enough to handle a wide-range of superimposed models. Explicitly representing and storing model, schema, and instance, along with being flexible in which is defined first, differs from most other approaches. In common use, metadata storage systems only represent two levels of information (i.e., the metadata schema and the metadata itself) and the schema must be defined prior to the metadata instance. The Meta Object Facility (MOF) [19] with the XML Metadata Interchange (XMI) and the Microsoft Repository [1, 2] with the Open Information Model are two such examples.

6. Contributions and Current Work

We presented an architecture for applications based on superimposed information, prototype implementations of generic components of that architecture, and the SLIMPad application built on top of them. The main components are the Mark Manager, the SLIM Store and the superimposed application itself.

The Mark Manager isolates the SLIM Store and the superimposed application from both the addressing modes for base information sources and the native applications that use that information. The Mark Manager has proven readily extensible—the amount of modification to a base application is small, plus the interface of marks to the rest of the system remains fixed. For the SLIM Store, our design decision was towards maximum flexibility, with data model as well as schema being selectable and explicitly represented. The trade-off for this flexibility was space efficiency of the data and the cost of interpreting manipulations on SLIM Store data. However, this tradeoff seems justified, as we expect the volume of superimposed information to be a fraction of the base data. To shield the superimposed application itself from this multi-level approach, we provide an object-oriented DMI customized to the particular schema of the superimposed application. Our overall assessment is that the architecture "works": It allowed parallel development and extension of the Mark Manager, SLIM Store, and SLIMPad. We expect to test it further in other superimposed information applications.

We also described the construction of SLIMPad, a particular superimposed application built according to the architecture with our generic components. SLIMPad supports creation of and access to digital "bundles" of "scraps" of information that are connected to information elements in base sources. SLIMPad represents something of an extreme point in information system design. Much of the semantics it captures is implicit—the application doesn't attach any special meaning to labels on bundles and scraps, nor their nesting and juxtaposition. Thus SLIMPad is limited in operations or services it can provide on the information it manages. On the other hand,

SLIMPad is minimally constraining on its users: There is no "schema-first" requirement, there are no required fields to fill in and no typing constraints. Yet our initial feedback from potential users of a SLIMPad-like application in the medical domain is that it provides useful functionality: selection and regrouping of information elements, plus the ability to re-establish context for the selections.

In terms of on-going work on the architecture, we are considering additional behavior on marks that would be available to superimposed application builders, such as "extract content" and "display in place." Such an extension will require new mark modules for an existing mark type. With the SLIM Store, we have been investigating the automatic generation of customized data manipulation interfaces from high-level specification, using techniques from domain-specific languages. We are also considering augmenting such interfaces with query capabilities, in addition to the current navigational access. In applications of our SLIM Store technology beyond SLIMPad, some data sets are quite large and we are developing alternative implementation mechanisms. We are also developing capabilities for cross-schema and even cross-model mapping of superimposed information.

For the SLIMPad application itself, there are extensions contemplated to its information model that correspond to real world manipulations of bundled information. These include annotations on scraps, linking among scraps and templates for bundles. Our current direction is to use SLIMPad as the basis for a task-specific tool prototype in the medical domain that we can test with clinicians. A likely task area is supporting the transfer of "current situation" awareness for hospital patients when one doctor is taking over rounds for another, such as on weekends.

7. Acknowledgement

We gratefully acknowledge Phil Bernstein's help in herding this paper into shape.

8. References

[1] P.A. Bernstein, et al. Microsoft Repository Version 2 and the Open Information Model. *Information Systems* 24(2), pp. 71-98, 1999.

[2] P.A. Bernstein, T. Bergstraesser, Meta-Data support for data transformations using Microsoft Repository. *IEEE Data Eng. Bulletin* 22(1), pp. 9-14, March 1999.

[3] M. Biezunski, M. Bryan, S. Newcomb, eds. ISO/IEC 13250, *Topic Maps*, http://www.ornl.gov/sgml/sc34/-document/0058.htm.

[4] S. Bowers and L. Delcambre. Representing and transforming model-based information. Euro. Conf. on Digital Libraries, Wkshp. on the Semantic Web, Lisbon, Sept. 2000.

[5] D. Brickley, R.V. Guha, eds. Resource Description Framework Schema (RDFS), W3C Proposed Rec. 03-Mar-1999, http://www.w3.org/TR/PR-rdf-schema/.

[6] L. Delcambre, D. Maier. Models for superimposed information. *Advances in Conceptual Modeling ER '99*, LNCS 1727, pp. 264-280, Paris, Nov. 1999.

[7] S. DeRose, E. Maler, D. Orchard, B. Trafford, eds. XML Linking Language (XLINK), W3C Working Draft 21-Feb-2000, http://www.w3.org/TR/2000/WD-xlink-20000221.

[8] P.N. Gorman, et al. Bundles in the wild: Tools for managing information to solve problems and maintain situation awareness. *Library Trends* 49(2), pp. 266-289, Fall 2000.

[9] Edwin Hutchins. *Cognition in the Wild.* MIT Press, 1995.

[10] E. Hutchins and T. Klausen. Distributed cognition in an airline cockpit. In *Cognition and Communication at Work*, New York: Cambridge University Press, pp. 15-34, 1996.

[11] IBM ActiveNotebook. http://www.alphaWorks.ibm.com/tech/activenotebook.

[12] O. Lassila and R.R. Swick, eds. Resource Description Framework (RDF) Model and Syntax Specification, W3C Rec. 22-Feb-1999, http://www.w3.org/TR/REC-rdf-syntax.

[13] D.T. Lawton, I.E. Smith. The Knowledge Weasel hypermedia annotation system. *Hypertext 1993 Proceedings*, pp. 106-117, Seattle, Nov. 1993.

[14] W.-S. Li, et al. PowerBookmarks: A system for personalizable web information organization, sharing, and management. *Proc. SIGMOD 1999*, pp. 565-567.

[15] W.E. Mackay. Is paper safer? The role of flight strips in air traffic control. *ACM Transactions on Computer-Human Interaction* 6(4), pp. 311-340, 1999.

[16] D. Maier and L. Declambre. Superimposed information for the Internet. *WebDB'99*, pp. 1-9, Philadelphia, June 1999.

[17] Microsoft Monikers. http://msdn.microsoft.com/library/-psdk/com/monikers_1xpv.htm.

[18] S.H. Myaeng, et al. A digital library system for easy creation/manipulation of new documents from existing resources. RIAO2000, Paris, Apr. 2000.

[19] Object Management Group. Meta Object Facility (MOF) Spec. OMB Doc. ad/99-09-04. http://www.omg.org/cgi-bin/doc?ad/99-09-04.

[20] T. Phelps and R. Wilensky. Multivalent annotations. *Research and Advanced Technology for Digital Libraries* (ECDL '97), LNCS 1324, pp. 287-303, Pisa, Sept. 1997.

[21] M. Rosheisen, C. Mogensen, T. Winograd. Shared web annotations as a platform for third-party value-added information providers. STAN-CS-TR-97-1582, Stanford Integrated Digital Library Project, Nov. 1997.

[22] J. Rumbaugh, I. Jacobson, G. Booch. *The Unified Modeling Language Reference Manual*. Addison Wesley, 1999.

[23] ThirdVoice. http://www.thirdvoice.com.

[24] M, Weaver. SLIIM-ML. Oregon Graduate Institute internal memorandum, Mar. 2000. http://www.cse.ogi.edu/~mweaver/papers/slim-ml.pdf