情 報 メ デ ィ ア 39－6
（2001．3．23）

An Information Space Architecture: Topica Framework

Yuzuru Tanaka, and Jun Fujima
Meme Media Laboratory, Hokkaido University, Sapporo, 060-8628 Japan
tanaka@meme.hokudai.ac.jp, fujima@meme.hokudai.ac.jp

Abstract. With the growing need for interdisciplinary and international availability, distribution and exchange of intellectual assets including information, knowledge, ideas, pieces of work, and tools in re-editable and redistributable organic forms, we need new media technologies that externalize scientific, technological, and/or cultural knowledge fragments in an organic way, and promote their advanced use, international distribution, reuse, and re-editing. These media may be called meme media since they carry what R. Dawkins called "memes". An accumulation of memes in a society forms a meme pool that functions like a gene pool. Meme pools will bring about rapid accumulations of memes, and require new technologies for the management and retrieval of memes. This paper reviews our R&D on meme media, meme pools, and proposes a new framework called 'Topica' for organizing and accessing the huge accumulation of intellectual resources in our societies.

1. Introduction

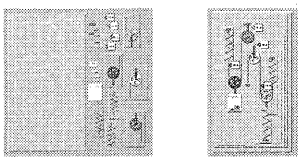
If we look back over the last three decades of computer systems, we can summarize them as follows. In the 1970s, we focused on the integrated management of enterprise or organization information, and developed database technologies. In the 1980s, we focused on providing an integrated environment for personal information processing and office information processing, based on the rapid development of personal computers and workstations that began in the late 1970s. The object-orientation paradigm played an essential role in developing graphical user interface and unified treatment of data, texts, figures, images movies, and programs. In the 1990s, we focused on the publication and browsing of intellectual assets, based on the rapid development of WWW and browser technologies.

One of the possible scenarios for the coming decade may be the further growing need for interdisciplinary and international availability, distribution and exchange of intellectual assets including information, knowledge, ideas, pieces of work, and tools in re-editable and redistributable organic forms. We need new media technologies that externalize scientific, technological, and/or cultural knowledge fragments in an organic way, and promote their advanced use, international distribution, reuse, and re-editing.

These media can carry a variety of intellectual assets. A media object denotes such a medium with some intellectual asset as its contents. Such media objects can replicate themselves, recombine themselves, and be naturally selected by people reusing them. We call them 'meme media' since they carry what Richard Dawkins called 'memes'. In his book, "The Selfish Gene" [1], Dawkins suggested provocatively that ideas (he called them memes) are like genes and that societies have meme pools in just the same way as they have gene pools. Whereas genes exist in a physical environment, 'memes' exist within a society. A fundamental and necessary framework for the growth and distribution of 'memes' is a 'meme pool'. A 'meme pool' is an accumulation of 'memes' in a society and functions like a gene pool. 'Meme media', together with a 'meme pool', provide a framework for the farming of knowledge. When economic activities are introduced, a 'meme pool' becomes a 'meme market' where providers and distributors of 'memes' should be able to carry out their business without prohibiting the replication, re-editing and redistribution of 'memes' by users.

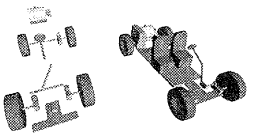
Based on these predictions, we have been conducting research and development on 'meme media' and 'meme market' architectures since 1987. We developed 2D and 3D meme media architectures 'IntelligentPad' and 'IntelligentBox' respectively in 1989 and in 1995 [2~7], and have been working on their meme-pool and meme-market architectures [8, 9], as well as on their applications. IntelligentPad represents each object as a pad, i.e., a card-like visual object that you can directly manipulate on the display screen (Figure 1), whereas IntelligentBox represents each object as a box, i.e., a 3D visual polyhedron object with direct manipulability (Figure 2). Both of them allow us to directly combine different objects on the screen to compose new objects. In the IntelligentPad architecture, you can paste a pad on

another pad, whereas, in the IntelligentBox, you can embed a box in the local coordinate system defined by another box. In each of these cases, the former object becomes a child of the latter. Each of pads and boxes exports its functional linkage capability as a list of slots. When you combine an object with another, you can functionally connect the child object with one of the slots defined by the parent. Figure 2 shows a car composed of primitive boxes. When a user rotates its steering wheel, the steering shaft also rotates, and the rack-and-pinion converts the rotation to a linear motion. The cranks convert this linear motion to the steering of the front wheels. This composition requires no additional programming.



(a) primitive pads (b) a composite pad

Fig. 1 An example pad composition.



(a) primitive boxes. (b) a composite box.

Fig. 2 An example box composition.

Based on these meme media architectures, our group and our collaborators from academia and industry have developed a large variety of applications. Some of them include PIM (Personal Information Management) systems, CAI systems, multimedia KIOSK systems, GISs (Geographical Information Systems), digital archive and interactive access of Kyoto cultural heritage, international exchange and distribution of nuclear reaction data and their analysis tools, interactive visualization of a cDNA database, and interactive visualization and simulation of electromagnetic fields caused by cellular phones of different shapes.

The meme media and meme pool architectures will bring about a rapid accumulation of memes in our societies, which will require a new way of organizing and accessing them. No conventional information organization method, such as table-based, hierarchical, or indexed, is suitable for organizing and allowing access to a huge number of heterogeneous intellectual assets. The situation here is similar to the management and access of commodities in our societies. While commodities of the same type can be managed by a single database, there are so many different types that consumers cannot tell either which commodity belongs to which type, or which database manages which type. To solve this problem, we used to use documents or spaces to arrange information about mutually related commodities. Examples include catalogs, stores, department stores, malls, and towns. Here we propose a new framework for organizing and accessing intellectual assets. This framework uses documents to contextually and/or spatially select and arrange mutually related assets. Examples of such documents may include figures, images, movies, maps, and any combinations of them. These documents, as well as their component assets, are all represented as meme media objects. Therefore, these documents, together with related assets, may also be arranged in other documents, which forms a complex web of such documents.

2. Intellectual Assets on Meme Media

IntelligentPad and IntelligentBox have versatile application fields. They have a capability of covering all kinds of client applications using 2D and 3D graphical representations. Each application may require the development of new primitive pads or boxes. Figure 3 shows PIM tools developed as composite pads, while Figure 4 shows a GIS using IntelligentPad and a legacy GIS database engine. Nigel Waters, a professor in the Department of Geography at the University of Calgary, proposed a GIS application of IntelligentPad [10] in which a map display, a traffic simulation model, a video image of an intersection, and a display in graph form are all represented as mutually interacting pads. Such a GIS is not only a great pedagogical device, but is also invaluable for planning.

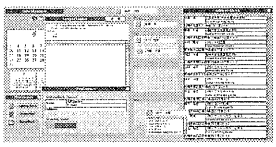


Fig. 3 PIM (Personal Information Management) tools developed as composite pads.

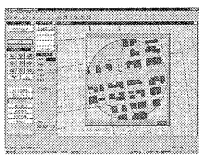


Fig.4 a GIS (Geographical Information System) using IntelligentPad and a legacy GIS database engine.

Seigo Matsuoka of Editorial Engineering Laboratory (EEL) Inc. applied IntelligentPad to the

production of 'The Miyako', a digital archive system for Kyoto cultural heritage (Figure 5). It stores all the multimedia contents in a relational DBMS, and uses IntelligentPad for its front-end interface to provide full interactivity so that users can navigate through the huge contents library using various types of association search based on Japanese culture.

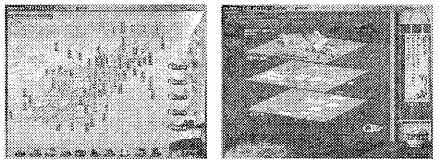


Fig. 5 A digital archive system 'The Miyako' using IntelligentPad.

As to IntelligentBox, we have developed two important generic application frameworks; one for interactive database visualization and the other for interactive scientific visualization. We have been collaborating with Takashi Gojobori's group at National Institute of Genetics to develop an interactive animation interface to access cDNA database for the cleavage of a sea squirt egg from a single cell to 64 cells (Figure 6). The cDNA database stores, for each cell and for each gene, the expression intensity of this gene in this cell. Our system animates the cell division process from a single cell to 64 cells. When you click an arbitrary cell, the system shows the expression intensity of each of a set of genes specified in advance, as shown in the left lower part of this figure. You may also arbitrarily pick up three different genes to observe their expression intensities in each cell. The expression intensities of these genes are associated with the intensities of RGB color components to highlight each cell in the cleavage animation. Keeping this highlighting function active, you can advance or step back the cell-division animation. The cDNA database is stored in an Oracle DBMS, which IntelligentBox accesses using Java JDBC.

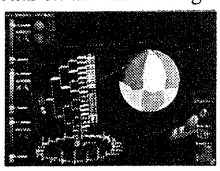


Fig. 6 An IntelligentBox application to a 'biosimulator' accessing cDNA database

For interactive scientific visualization, IntelligentBox provides a generic linkage mechanism with the AVS system. This allows us to define a box as a program module of AVS, so that combination of such boxes defines a composition of an AVS program, and the manipulation of such a box changes parameter values of its corresponding AVS program module. These allow us to define a virtual laboratory in which we can construct a scientific simulation world through direct manipulation of previously constructed components, directly manipulate objects in this world to change situations, and interactively observe the simulation result in this world. Figure 7 shows a virtual laboratory for experimenting with the antenna of a cellular phone. Users can directly change the location and the length of the antenna to observe the changes of the radiation pattern, the electric field, and the surface current on the phone body. The system uses NEC2 as a numerical computation solver, which is invoked through AVS.

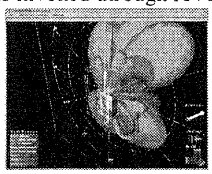


Fig. 7 Virtual Lab System using IntelligentBox technologies.

All these application pads and boxes are decomposable. Users can reuse some of their components, or customize them by replacing some components with others or adding new ones.

3. Meme Pool and Meme Market Architectures

In order to make pads and boxes work as memes in our societies, we need a worldwide publication repository that works as a meme pool. The Piazza architecture allows us to define such repositories of pads over the Internet (Figure 8). Each repository is called a pizza. You can drag-and-drop pads between any piazza and your own local environment. You can easily define and open your piazza over the Internet, and register its entry-gate pad to any other public piazza. People accessing the latter pizza can access your new piazza by double-clicking this entry-gate pad. Users of web browsers have to ask web page owners by sending, say, an e-mail for including his or her information in another's web page, or for spanning links from another's web page to his or her page. This is similar to the situation between tenants and owners. The Piazza architecture, on the other hand, provides a large public marketplace for people to freely open their own stores or galleries of pads.

The Piazza architecture consists of a Piazza server and a Piazza browser. A Piazza browser is represented as a pad, and supports browsing among different 'piazzas'. Each piazza is associated with a

file managed by a remote Piazza server. Pads can be drag-and-dropped to and from the currently accessed piazza, to upload and download pads to and from the associated remote server file. When a piazza is opened within a Piazza browser, all the pads registered at the associated server file are immediately downloaded onto this piazza and become available. An entrance link to a piazza is also represented as a pad, and can be put on another piazza to define a link. Users are welcome to install their own Piazza servers anywhere, anytime, and to publish their piazzas. Piazza enables end users to open their own gallery of pads on the Internet, or to exhibit their pads in some other private or public space. Such pad galleries work as flea markets, shops, shopping centers, community message boards, community halls, or plazas.

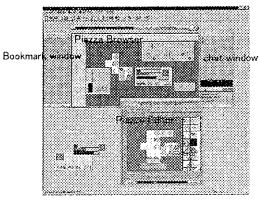


Fig. 8 A piazza with registered pads (top), and a piazza editor to define a new piazza (bottom).

4. Topica for Organizing and Accessing Intellectual Assets

4.1 Organization and access of intellectual assets

All the example composite pads and boxes in preceding chapters are subject to international distribution, exchange, and reuse. These data and tools, as well as their documents in pad or box forms, serve as intellectual assets in our societies. The meme media and meme pool architectures will rapidly increase their variety, and form their huge accumulation.

Now we consider how to manage and access a huge accumulation of intellectual assets represented as pads or boxes. Here we consider only pads, but our conclusions are also applicable to boxes. Let us first consider if databases can manage pads. If the platform has the pad definition code and all the necessary DLLs, the storage of a composite pad only needs to store its exchange format representation; no other information needs to be stored. The exchange format representation of a composite pad includes two kinds of information. One is the form information that describes what kinds and sizes of component pads are used, how they are geometrically pasted, and which slot is used in each connection between component pads. The other is the state information of this pad. The state information needs to be sufficient to specify the current values of all of its internal variables whose values are not specified by its form information. Composite pads with the same form information but with different states are said to share the same form. Without loss of generality, we can assume that the state information has a record type, i.e., it can be represented as a list of attribute-value pairs for the ordered attribute set that is determined by each form.

If we only have to manage a large number of pads of a few different forms, we can keep the form information outside the databases; we only need to store the state information of pads in the databases. Such a database is called a form base. If the state information of a record type has only atomic and simple values for its attributes, we can use a relational database system to store these pads. If some attributes allow variable length data, continuous data such as movies and sounds, or complex data such as compound documents and other relations, we can use an extended relational database system or a structural OODB system. In this case, we can even deal with a composite pad storing other composite pads in some of its state attributes.

Pads representing various intellectual assets accumulated in our societies, however, have a large number of different forms. While we may store a group of pads of the same form in a single database relation, we have to manage a huge number of different relations together with the same number of different forms.

The situation here is similar to the management and access of commodities in our societies. Different from standardized prefabricated parts that are usually managed by databases, commodities in our societies have a huge variety and no common attributes to describe them, which makes it difficult to manage them with databases. While commodities of the same type can be managed by a single database, there are so many different types that consumers cannot tell either which commodity belongs to which type, or which database manages which type. Types are usually defined by producers, and not always directly related to functions, uses, or appearances that consumers can identify. To solve this problem, we typically use documents or spaces to arrange information about mutually related commodities for ease of access. Examples of such documents are catalogs published by producers or independent publishers, advertising brochures from producers or stores, books, periodicals, and

newspaper articles referring to commodities. Catalogs adopt various different criteria in the selection and arrangement of commodities. Books, periodicals, and newspapers may refer to each other. Examples of commodity-organizing spaces include shops, department stores, malls, and towns. The first three use consciously planned selections and arrangements, while the selections and arrangements in towns evolve emergently. Shops are nested in malls and department stores, which are again nested in towns.

Let us consider one more example. 'The Trinity' is one of the most popular themes of Christian paintings. Each painting with this theme includes the images of the Father, the Son, and the Holy Spirit. Suppose you have a collection of these three images extracted from a large number of paintings on the Trinity. Our question here is where to store this collection so that we or even other people can access this collection in a future. You may think that we can define a relation in a database to store this collection. This relation has three attributes, the Father, the Son, and the Holy Spirit. Each tuple is a triple of file pointers, pointing to three images extracted from the same painting. This solution, however, does not tell where to memorize the fact that this newly created relation represents the collection of three images from a large number of paintings on the Trinity. We have to deal with a huge number of different concepts as well as relations among them. 'The trinity' is only one of them. A potential solution in this example may be to store this collection in association with the article on the Trinity in some encyclopedia.

4.2 Topica Framework

Based of the above observations, here we propose a new framework for the organization of and access to intellectual assets represented as pads. This framework uses documents to contextually and/or spatially select and arrange mutually related intellectual assets. Such documents may be texts, images, figures, movies, maps, or compound documents consisting of various multimedia components. These documents as well as these intellectual assets are all represented as pads. Therefore, these documents may be also arranged together with related assets on other documents.

We call this framework 'Topica', named after Aristotle's *Topica*. In the Topica framework, documents to arrange assets are called topica documents. Each topica document is a pad that displays a document and stores relations among some other topica documents and/or some pads. Such a document is represented by an XHTML text, with some slot definitions. Relations in a topica document are called 'topica tables', and may be defined by tables, or by queries that may access local or remote databases, XHTML texts defining other topica documents, or relations defined in other topica documents. A topica document has some areas through which users can store and retrieve other topica documents, pads, or character strings; we call these areas on a topica document 'topoi'. Each topos is basically associated with an attribute of the topica tables stored in the topica document. Each attribute within a topica table may take as its value a character string, an exchange format representation of a pad, or a URI identifying a topica document or a pad stored in a local or remote file.

A topos of a topica document is either a geometrically specified area of this document or a tagged text string in the XHTML document that is viewed by this topica document. Figure 9 shows an XHTML document on 'the Trinity' in Christianity, where a special kind of tag is used to specify that the three phrases 'the Father', 'the Son' and 'the Holy Spirit' in this article, together with the title 'Trinity, The', work as four topoi of this topica document, which stores a relation among the images of the three depicted within each of a number of paintings of the Trinity. Instead of directly storing images, the relation stores URIs of these image files.

```
<?xml version="1.0"?>
<html
  xmlns="http://www.w3.org/..."
  xmlns:topica="..." >
  <head>
    <title>Trinity</title>
    <topica:table>
      <tuple>
        <father>file://C:/pub/trinity/father1.jpg</father>
        <son>file://C:/pub/trinity/son1.jpg</son>
        <spirit>file://C:/pub/trinity/spirit1.jpg</spirit>
      </tuple>
      ...
    </topica:table>
  </head>
  <body>
    <h1><topica:topos name="trinity"
      ref="//tuple/trinity/text()">Trinity</topica:topos>,
      The</h1>
    <p> The central ... in Three Persons,
      <topica:topos name="father" ref="//tuple/father/text()">
      the Father</topica:topos>,
      <topica:topos name="son" ref="//tuple/son/text()">
      the Son</topica:topos>, and
      <topica:topos name="spirit" ref="//tuple/spirit/text()">
      the Holy Spirit</topica:topos>. ..
    </p>
    ...
  </body>
</html>
```

Fig. 9 An XHTML definition of a topica document on the Trinity.

We can use a topica viewer pad to view the corresponding topica document as a pad. The topica viewer pad is basically the Microsoft Internet Explorer (IE) wrapped by a pad wrapper. It has extended IE to perform topoi functions. Topica documents may also provide some slots, which can be easily defined by using special tags in their XHTML definitions.

Figure 10 shows the topica document of the XTML definition in Figure 9, a selector popped up by double-clicking the 'Father' topos, and the selection of one candidate within this selector, popping up the corresponding image. This selection automatically influences the information available through other topoi; the clicking of 'son' topos now pops up a selector showing only one candidate. All these images also work as topica documents. Each image of a whole Trinity painting includes three topoi respectively covering the Father, the Son, and the Holy Spirit, and a topica table that refers to the topica table in the original Trinity article.

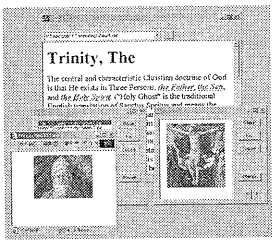


Fig.10 A topica document on the Trinity defined by the XHTML text in Fig. 9.

Topoi are different from Xlinks [11] in the following two respects: Topoi on the same topica document are related with each other by the topica table stored in this topica document. Secondary, you may drag-and-drop new topica documents into some topoi to update the topica table.

4.3 The application horizon of the topica framework

Figure 11 shows the management of invitation letters using a topica document. Invitation letters from the same person for the same category of purposes may share the same letter template, which he or she can reuse repeatedly to generate such letters by filling in the blanks. This topica document, on one hand, works as such template; underlined italicized strings may be rewritten for different letters. The same topica document, on the other hand, works to store and manage all the letters created using this template; the underlined italicized strings work as topoi. When clicked, each topos pops up a selector showing all the candidate strings filling in this placeholder; a selection of one of them replaces the current string, and rewrites the letter. This topica document has another topos to store resumés sent from invitees; an instantiation to some specific invitation letter also instantiates this topos to pop up the resumé of the selected invitee. This resumé also works as a topica document with some topoi. When you send invitation letters, you may also send the template for a resumé.

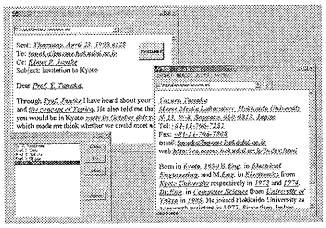


Fig. 11 A topica document for the management of invitation letters together with invitees' resume.

Suppose you have presented talks at many conferences in the past. For each conference, you have files of the call-for-paper mail, the submitted paper manuscript, the letter of acceptance with reviewers' comments, the camera-ready manuscript, the conference program, and the Power Point presentation. With our conventional file directory system, you have two alternative ways to store these files. You may either define an independent folder for each conference, or define six different folders for six different file categories. In the first case, you cannot scan through all the files of the same category. In the second case, you cannot jump from one file to another of a different file category, but of the same conference. The topica document in Figure 12 solves this problem. Each folder in this directory corresponds to one of the six file categories, and works as a topos. Its double-click pops up a selector that looks like another file directory listing all the files of this category. A selection of one file in this selector determines the corresponding conference, and restricts every other topos to show only one file of the same conference.

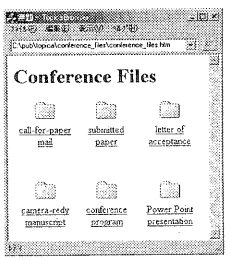


Fig. 12 A topica document that works as a file directory with 6 categories of files.

Each topica document may play three different roles. First, it works as it is. Second, it may work as a template in such a way as shown in the invitation-letter example. Third, it works as a schema of the stored topica table; you may use a topica document to specify a query to the topica table. The last role will be detailed in the following. To distinguish these three different roles of the same topica document, we have introduced three modes for each topica document; they are the document mode, the template mode, and the schema mode. You can change the mode of a topica document by popping up the right button menu of this document. Unless otherwise specified, each topica document is in its document mode.

If we restrict every topica document to store a single relation with a single tuple, then each topos works as an anchor to another topica document. In this way one could replicate the standard linking behavior of a WWW page. However, topoi can provide the additional behavior of allowing update of their underlying values by users other than the topica document owner. Update of topica document relations through direct manipulation operations will be reported elsewhere.

4.4 Queries over the web of topica documents

The Topica framework provides a unified approach for organizing and accessing local and/or remote files, databases, conventional web documents including search engines and portal sites, and topica documents over the Internet. In addition, the framework allows us to describe queries in XML-QL that, by navigating through these different types of information, quantifying properties of some documents on the navigation path, and picking up selected assets on the way, can construct the XHTML documents and relations of new topica documents. Figure 13 shows an example XML-QL query.

This query accesses a topica document identified by the variable $myReferenceBook whose value is specified elsewhere, and retrieves all the books from its topos named 'encyclopedia'. Then it selects 'Christianity' for the 'topics' topos defined on each title page of these books (i.e., encyclopedias) to retrieve all the articles on Christianity from each of these books. Then it searches these articles for those with 'Trinity' as its header, and retrieves all the images of 'the Father' from each of these article topica documents. Finally, it generates a new topica document storing the collection of these retrieved images in its 'image' topos.

```
CONSTRUCT
<topicaDocument>
  <style ref="http://ca.meme.hokudai.ac.jp/scrapbook.xsl"/>
  <contents>
    This is a collection of <topos name="father"
    ref="//image/text()"> the Father images</topos>
    in the paintings of Trinity.
  </>
  <topicaTable> {
  WHERE
      <topicaTable> <tuple>
          <encyclopedia>$encyclopedia</>
      </></> IN $myReferenceBook,
      <topicaTable><tuple>
          <topics>Christianity</>
          <articles>$articles</>
      </></> IN $encyclopedia,
      <html>
          <head><title>Trinity</></>
          <body>
           <topicaTable><tuple><father>$father
           </></></></>
          </>
      </> IN $articles
  CONSTRUCT
      <tuple><image>$father</></>
  } </>
</>
```

Fig. 13 An example XHTML to create a new topica document by navigating through existing ones.

The XML-QL description above, however, has a serious problem. Is it reasonable to assume that the user knows all the topica tag names necessary to specify this query? Obviously the answer is 'No'. However, he or she can navigate through topica documents along a single path consistent with this query. Figure 14 shows a history of such a navigation starting from a file directory 'myReferenceBook', and ending with an article on 'Trinity, The'.

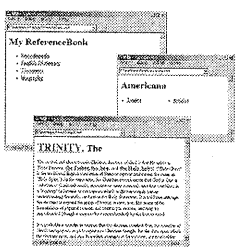


Fig. 14 A single path navigation that is consistent with the query in Fig.13.

By changing these topica documents to the schema mode, you can specify a query as shown in Figure 15. This visual query specification basically exploits the QBE (Query-By-Example) convention. Every underlined topos value works as a variable, which may specify either a text string or a topica

document. You may specify these variables either on a topica ocument in its schema mode or on a topos selector window. In it schema mode, every topica document has one additional topos at the top left corner, which is used to specify the URIs of topica documents sharing the same schema with this topica document. This query searches for all the possible navigation paths starting from the directory 'myReferenceBook', and ending with an article including the string 'Trinity' in its header. The query also specifies an output topica document with the phrase 'the Father images' specified as a topos, and equates this topos with 'the Father' topos of the 'Trinity, The' topica document by using the same variable for these topoi. All the topica documents in the condition part of this query should be interpreted as schemas, while the one in the output part defines a template. The query in Figure 15 specifies the same query as the XML-QL query in Figure 13.

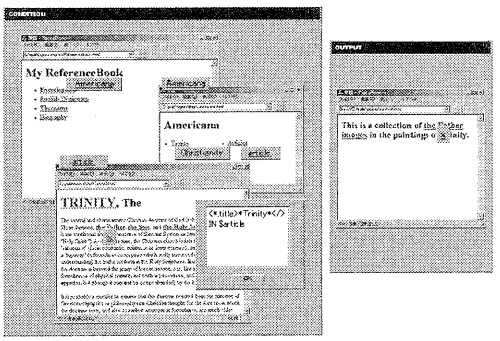


Fig.15 Visual specification of the same query specified in Fig.13.

The visual query specification using topica documents in their schema mode assumes that topica documents of the same category, or on the same topics use the same topica tag names in their definitions. Such convention of using the same tag names spreads among people either through standardization efforts, or through the extensive replication and distribution of the same topica document among people to reuse its contents, style and/or schema in their production of new topica documents. Topica documents as templates and/or schemas will also become intellectual properties, and provide new business opportunities.

5. Concluding Remarks

Meme media and meme market system architectures work as the enabling technologies for interdisciplinary and international availability, distribution and exchange of intellectual assets including information, knowledge, ideas, pieces of work, and tools in re-editable and redistributable organic forms. Pads and boxes as meme media objects allow us to re-edit and internationally redistribute a huge variety of tools and documents including CAI tools, interactive animation objects, PIM tools, GIS tools, digital archives and their access tools, database visualization tools, virtual lab. tools, meme-pool organization and access tools, and topica sheet documents. Meme media and meme market system architectures will significantly accelerate the evolution of memes in our societies, which will lead to a need for new ways of organizing and accessing their huge accumulation. The Topica framework provides a unified framework for organizing and accessing local and/or remote files, databases, conventional web documents, and topica documents over the Internet. It uses documents to contextually and/or spatially select and arrange mutually related intellectual assets distributed over the Internet.

References

[1] R. Dawkins. *The Selfish Gene*. Oxford Univ. Press, Oxford, 1976.

[2] Y. Tanaka, and T. Imataki. IntelligentPad: A Hypermedia System allowing Functional Composition of Active Media Objects through Direct Manipulations. In *Proc. of IFIP'89*, pp.541-546, 1989.

[3] Y. Tanaka, A. Nagasaki, M. Akaishi, and T. Noguchi. Synthetic media architecture for an object-oriented open platform. In *Personal Computers and Intelligent Systems, Information Processing 92, Vol III*, North Holland, pp.104-110, 1992.

[4] Y. Tanaka. From augmentation media to meme media: IntelligentPad and the world-wide repository of pads. In *Information Modelling and Knowledge Bases, VI* (ed. H. Kangassalo et al.), IOS Press, pp.91-107, 1995.

[5] Y. Tanaka. A meme media architecture for fine-grain component software. In *Object Technologies for Advanced Software*, (ed. K. Futatsugi, S. Matsuoka), Springer, pp.190-214, 1996.

[6] B. Johnstone. DIY Software. *New Scientist*. Vol.147, No.1991:26-31, 1995.

[7] Y. Okada and Y. Tanaka. IntelligentBox:a constructive visual software development system for interactive 3D graphic applications. *Proc. of the Computer Animation 1995 Conference*, pp.114-125, 1995.

[8]Y. Tanaka. Meme media and a world-wide meme pool. In *Proc. ACM Multimedia 96*, , pp.175-186, 1996.

[9] Y. Tanaka. Memes: New Knowledge Media for Intellectual resources. *Modern Simulation and Training*, 1, pp.22-25, 2000.

[10] N. Waters. POGS: Pads of Geographic Software. *GIS World*, 8(11): 82, 1995.

[11] http://www.w3.org/TR/xlink/