ISSN 1872-4981

IDT

Volume 2, Number 2, 2008

Special Issue: Knowledge Visualization
Edited by Prof. Dr. Joachim Hasebrook

Intelligent Decision Technologies

An International Journal

IOS Press

KES

Intelligent Decision Technologies 2 (2008) 103–115
IOS Press

visuos™ – Strategies and user interface concepts for next generation knowledge work systems

Clemens Lango
via www.interactiondesign.de

Abstract. Today's knowledge workers have access to practically endless amounts of information resources. However, this information alone is of no use if it cannot be reasonably retrieved and organized. Today's software solutions are a collection of heterogeneous, inconsistent, separate environments that are not optimized for efficiency in the higher workflows typical of knowledge work.
visuos™ is a concept for an operating software for knowledge work. It covers various forms of access to knowledge spaces, including knowledge management, information retrieval, research and refinement, monitoring of knowledge resources, and automation and scheduling of tasks. It is a system designed for a broad target group – from novice to expert. Its modular structure allows users to adapt the system to their particular degree of experience and the individual form of access to knowledge spaces desired. The modules are integrated by a holistic conceptual model. This model was designed in order to allow effective workflows with minimum friction.
The benefits of the system include improved functionality, new workflows, a maximized target group, higher efficiency and productivity, and a higher degree of user confidence.

1. The visuospatial interfaces™ approach

visuos [3] follows the visuospatial interfaces paradigm evangelized by Clemens Lango as strong concept for interactive systems. It stands for interaction structures that offer a spatially consistent organization of a system's modules. "Cuts" that force the user to adapt to new orders that don't spatially refer to previously learned situations are cognitively demanding and require adaptation processes. These increase the risk of a user to feel cognitively overstrained and provoke operating errors.

Benefits of visuospatial interfaces for the user include:

- easy constitution of a conceptual model
- highly effective learning curve
- immediate feeling of familiarity with a system's structure
- natural feeling of secureness – "having the system under control"
- impression of controllability from the beginning of usage – also for non-computer affine target groups

2. Perfecting representational structures

The operating software developed in this project is a new user-centered approach for the knowledge worker and other users in hyperspace. The structure and the highly visual approach of the interface is intended to simplify many workflows of various approaches to knowledge spaces.

Some of the classical systems are limited because their interaction structure is too complex – far beyond the users' mental model of hyperspaces. Others are subject to the technological restrictions of processors and network capacity, or to insufficient indexing quality of content.

Apart from all these obstructions around hyperspaces, there are some fundamental qualities that, on

ISSN 1872-4981/08/$17.00

one hand, facilitate flexible forms of work, and on the other hand, go far beyond the cognitive abilities of the human being, as most systems do.

3. Narrowing the cognitive load to a human's cognitive measure

3.1. Endlessness of resources

The main quality of open systems is that they can handle a practically endless amount of content. Everyone can access everything and actively contribute to the system by publishing content of any kind. The advantage is a liberal, democratic system. The disadvantage is a massive overflow of information. Even when focusing on a narrow, very specific subject, there will be many resources available which to a large degree will appear interesting to a user.

It is very likely that the user is overwhelmed by, and perhaps enthusiastic about, the quantity of resources. The problem is that the user becomes dazed as he moves at a high speed through a massive amount of resources and loses sight of his original objective. A great deal of time passes by in an interval that is subjectively perceived as short. Hours fly by, and while at the end the user may feel that he had been entertained while browsing, in looking back, he also feels dissatisfied because he "lost" time without being productive or getting closer to the information he was originally looking for. This phenomenon can be described as asynchrony between real and virtual time. With today's systems, only users who have practiced effective strategies for quickly rating and filtering content can keep this phenomenon under control. In the system under consideration here, strategies should be found to lessen the risk of information overflow.

3.2. Limiting elements

The human mind can perceive only a limited number of elements at one time. In classical cognitive psychology, one speaks of the "magical number seven plus or minus two" [5]. As this value was targeted towards static environments in the context of the dynamic nature of interactive systems, the number of perceivable elements may be even smaller.

The system here proposed will by default restrict the number of elements represented. The criterion might be their relevance or hierarchy, for example. System levels not in the user's current focus should be omitted or at least be hidden.

3.3. Chunking

Chunking of elements reduces the absolute complexity of environments so that a mathematically higher number of elements can be presented to the user. By default, only a minimal amount of information about each element should be displayed – especially in overview situations. *Details* should be displayed *on demand* – either for all elements visible or only for the ones in focus.

3.4. Time management

Appropriate, easily perceivable solutions for representation and interaction with the *temporal dimension* have been conceived for all dynamic modules of the operating software, so that earlier states of the system can be easily recalled and handled. Work in knowledge spaces and hyper environments consists of various subprocesses. For each process an adequate functional model must be developed.

3.5. Optimizing productivity by developing one convergent meta-system covering all sub-processes

The problem with most solutions currently available is that they seriously address only one or two of these processes. There is no real meta-system integrating everything into one homogeneous solution, although only such a structure could minimize friction between the sub-modules and lead to a maximum degree of productivity within knowledge spaces.

Take the following as an example:

- a query from which
- the user follows one result,
- makes several decisions by linking to further connected elements,
- wants to recall an earlier state to choose a more promising trail,
- finds an interesting document and
- wants to sort this document into his local private workspace.

If these sub-processes were handled by different applications, there would be many distracting transitional steps to be performed by the user in addition to his actual intention. Furthermore, each of these systems speaks its own interaction language, which has to be learned. It is nearly impossible to learn the languages of three to five different systems and still have a fluent

workflow, because in each sub-process, the user would have to adapt to the corresponding interaction language. The reason for this lies in the difference between the interfaces of the applications and their mostly abstract forms of representation.

One single meta-system with an object-oriented visual representation of all elements and processes would allow the user to remain very close to his actual objectives without being distracted by multiple interaction structures and gaps between different systems.

4. The visual query

4.1. Definition of a query

In most search engines, queries are defined textually. To limit the number of results to a more specific and qualitative amount, advanced queries would have to be defined in the corresponding search engine's syntax, usually by entering combinations of terms and weighing them. The terms can often be defined with Boolean expressions (e.g., "and," "or"). This textual *query language* is quite abstract and is not very easy to edit and optimize. Only advanced users are capable of formulating more specific queries.

Visual Metrics Corporation [6] reports that "The average user enters 1.5 keywords per search engine search. [...] Use of structured, or 'Boolean' queries, while known to help obtain better search results, can be difficult and frustrating for some users to learn."

4.2. Representation of results

The representation of query results usually consists of text lists. In some cases, the relevance of the result is defined by additional indices. Often the user is presented with multiple pages of results that he has to sort through in order to find the relevant ones. The sorting criteria are usually the short descriptions or titles of the resources found. These text lists are highly abstract. The user receives a seemingly endless amount of feedback to process. He has to open multiple windows to check the relevance of the linked sites, and can easily be overwhelmed by the quantity given. When browsing through multiple linked sites the user can quickly lose track of the content presented.

4.3. Refinement of queries

In most cases the amount of results given is too broad. The user has to refine his query to narrow the results in order to get more specific ones. For most search engines, "refinement" means switching back to the textual "query formula" and changing or adding the terms or their Boolean combination. The difficulty is that there is no obvious, concrete connection between the results and the query formula, but rather a non-transparent gulf of relation. It is not evident which part of the query leads to unwanted results or how it could be optimized. So the only principle that can be applied is trial-and-error. The user has to modify his query again and again, running the risk of getting "lost in hyperspace." The user loses focus of the original aim because he is distracted by too much irrelevant information that seems to be "important."

> Being "lost" means many things ...
> not knowing where to go next;
> not knowing how to get where you want to go;
> not knowing where one is in relation to other places –
> disorientation; degradation of user performance [4].

4.4. Venn diagrams

John Venn, an English mathematician, developed the concept of Venn diagrams, which pictorially represent the relationships between sets based on Boolean logic (see Fig. 1).

Manually constructing these statements requires conscious reflection. In the *visual query* module of *visuos* [3], these are "calculated" by the system. The user either defines the Boolean statements visually and the formula is compiled automatically, or the user edits the formula and its visual representation is constructed by the system. Observing or picking sub-sets of the Venn diagrams is done unconsciously; understanding the sub-sets' meaning takes place implicitly.

4.5. Venn diagrams for a visual approach to Boolean queries

The visual query concept that forms part of this project goes one step further than the current solutions for queries, that is, towards a direct manipulative visual interface. A single graphical interaction model is used for definition of a query, representation of the results, and refinement of the query (see Fig. 2).

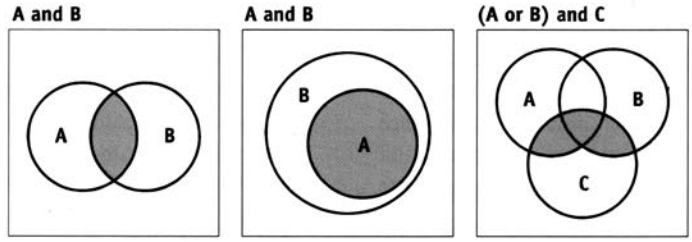

Fig. 1. Basic Boolean expressions.

Boolean operators and parentheses are replaced with a concrete visual, directly comprehensible graphical interaction model. Boolean expressions can be defined visually, and the user does not have to think in abstract mathematical formulas. Changes are applied by direct manipulation of the model.

Results of the query are displayed within the same conceptual model as the query. Compared to standard queries, this is more of a spatial map than a textual sequence. Elements of query results are contained in sets represented as circles. The complete result can be seen at a glance. No scrolling in endless sequences or switching between multiple screens is necessary. Query and result are merged in one graphical interaction model.

4.5.1. Refinement

The non-transparent, non-realtime gulf of relation between query and result is eliminated in this concept of a visual query model. The user gets detailed feedback on the distribution of hits. To refine the query, the part leading to the unsatisfying results is directly evident. The part of the result that does not suit the user's needs is simply modified or deleted.

Unlike today's standard search engines, there is no perceptually separated query formula that makes it difficult for the user to find out which part of it leads to unwanted results, or which part has to be optimized in order to find what he is looking for. These mechanisms significantly increase the efficiency of the information retrieval process. A substantial part of query definition and refinement has to do with controlling a result's quantity/quality ratio.

> The most critical problem in all queries is finding the right level of specificity for the subject query term(s). Too broad a keyword specification, and too many results are returned; too narrow a specification, and too few are returned [6].

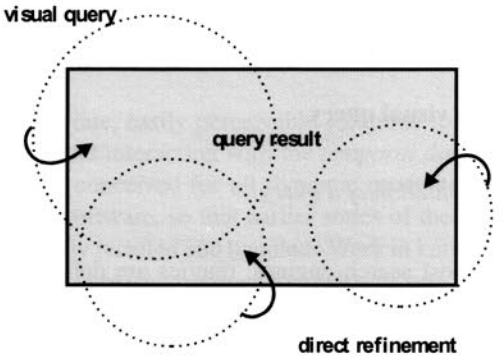

Fig. 2. Functional structure of the visual query in *visuos*.

5. The space

Spaces are visual representations of any class of public or private hypermedia resources. Their structure is determined by object classes (e.g., chapters, directory names) or sub-classes of the space and elements contained in it. There are, for example, content (documents or files), query, or space elements.

5.1. Examples of spaces

Systems that can be represented as spaces include:

- databases and information catalogs (e.g., list server archives, link lists/bookmarks, Yahoo)
- distributed file systems (classically accessed via ftp)
- Web site structures (classically represented as sitemaps)
- local file systems (e.g., hard disk, CD-ROM).

Classically all the classes of information spaces listed above are represented in different forms. Handling a variety of interaction paradigms while working across multiple differently represented, isolated systems is more of a task for advanced or expert users. Structurally seen, all those information spaces are very similar and could be described within one holistic represen-

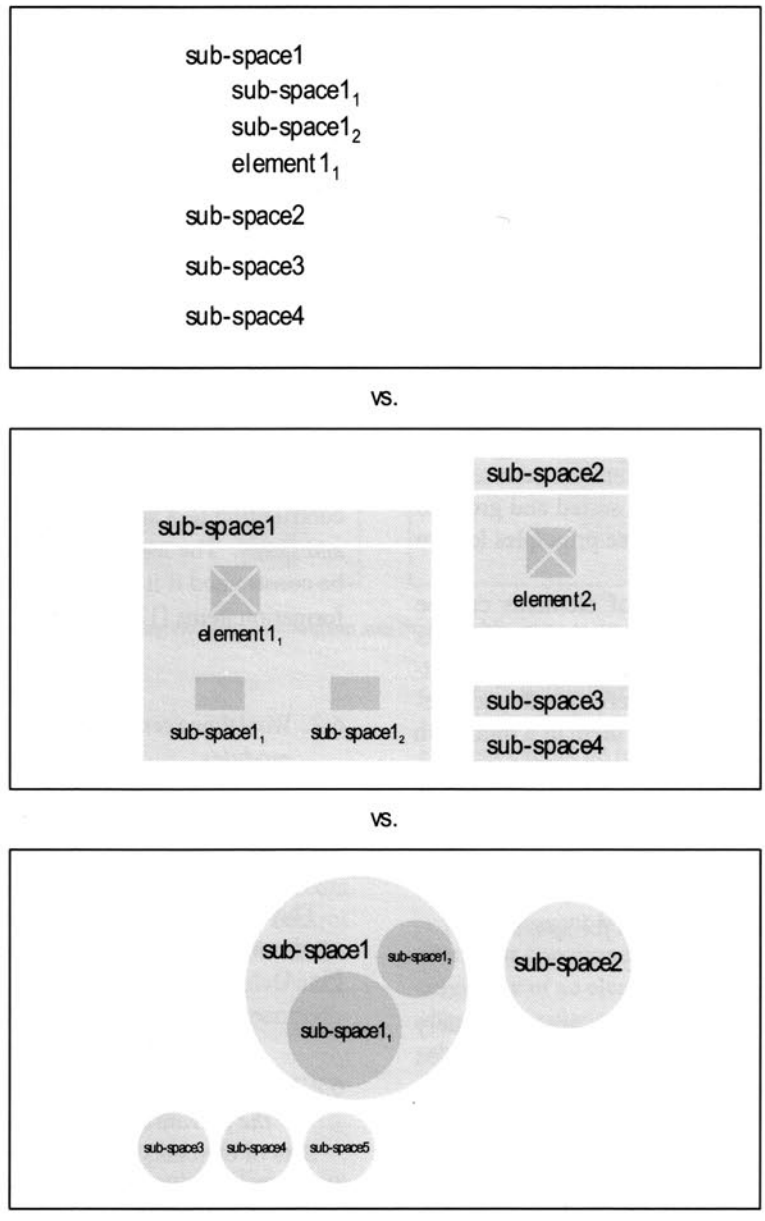

Fig. 3. Representation of hierarchical spaces in classic textual form, in a standard desktop OS and with Venn diagrams.

tational model. This strategy would minimize learning efforts and optimize working procedures which, in many cases, extend over multiple classes of information spaces.

That from a *traditional perspective* these spaces are separate systems with different forms of representation can only be explained in terms of independent historical, technology – not user-centered – driven development of functionality. From the *point of view of user experience,* those spaces are very closely related. The integration of these spaces' conceptual models will have nothing but advantages for the user and will enable completely new, more productive forms of work.

5.2. Venn diagrams for spaces

In the previous section, Venn diagrams were introduced as a conceptual model for visual queries. They can also be used for representation of the different classes of public and private hierarchical information spaces. The representation of these hierarchical spaces using Venn diagrams can be superior to classical rep-

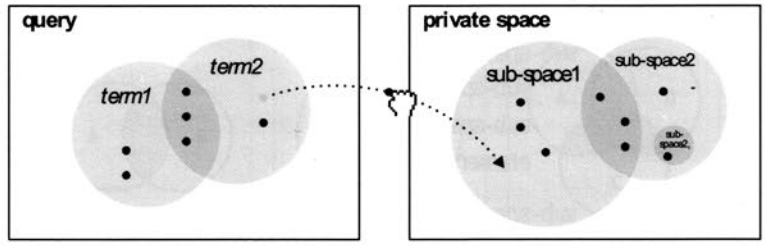

Fig. 4. Copying elements from a query result to the private space.

resentational forms. Information spaces and elements can not only be organized in hierarchical classes (like in classic systems). They can also be weighted. Weighting is made possible by the visual spatial representation of information spaces. Sub-elements (classes and elements) can be spatially placed, sorted and grouped into chunks within the space. These principles lead to characteristic cognitive maps.

The relevance or importance of elements can be represented by their position and their size. Objects can be visually identified and recalled. Classical text-based forms of representation of hierarchical spaces restrict the possible interaction with them to a minimum (Fig. 3).

6. Integration of spaces and queries

6.1. *One holistic conceptual model for spaces and queries*

From a traditional perspective, the access to query results in information retrieval (IR) tasks and the access to hypermedia differ significantly. But there have been attempts to integrate both forms of access.

> Hypermedia has tended towards the manual navigation of explicitly authored links, while IR has traditionally been oriented to the construction of analytic queries returning a ranked set of documents. Recent work in both disciplines has attempted to integrate these two approaches, although they still tend to appear as distinct phases of a user session. IR has gone a long way to incorporating browsing into essentially analytical strategies via the navigational interpretation of query results. If we consider hypermedia to have a different perspective, the issue is the incorporation of query into essentially browsing strategies [1].

This objective formulated by Cunliffe et al. is exactly what the integration of *spaces* and *queries* in *visuos* [3] accomplishes. Since the same conceptual model is used for *queries* and *spaces*, various workflows of a knowledge worker can be enormously simplified.

Such hybrid navigation tools are capable of overcoming some of the limitations of manual browsing and contributing to a smooth transition between browsing and query. The user's path through the system would be constrained if it relied on explicit links between information items [1].

6.2. *Workflow scenarios across query and space modules*

6.2.1. *Copying elements from a query result to the private space*

The user formulates a query, and drags and drops an element to his private space (local file system) (see Fig. 4).

6.2.2. *Copying complete sets from a query result to the private space*

Sets of elements resulting from a query are dragged (copied) to the private space (see Fig. 5).

6.2.3. *Copying elements or complete sets from a public space to the private space*

Elements or elements of a public publication are dragged (copied) to the private space (see Fig. 6).

Already these few examples give a first impression what highly productive new forms of usage as a combination of *queries* and *spaces* are made possible by the installation of one holistic graphical interaction model. They are not possible in classical applications, but rather represent completely new approaches to knowledge work while accelerating many working processes.

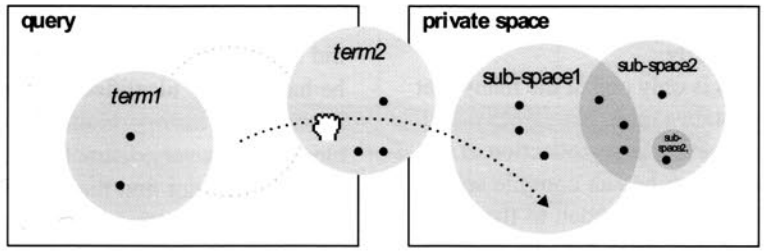

Fig. 5. Copying complete sets from a query result to the private space.

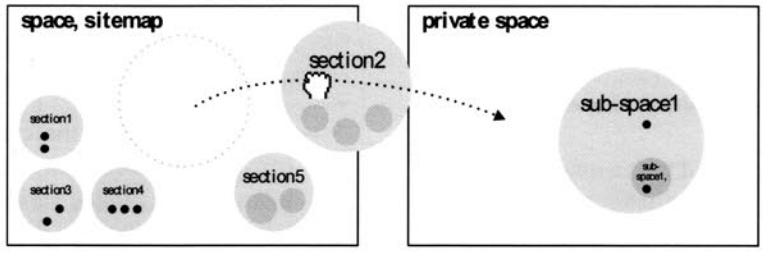

Fig. 6. Copying elements or complete sets from a public space to the private space.

7. Link classes and trails

7.1. Hypermedia vs. linear media

Hypermedia differ significantly from classical linear media. Whereas linear media, like a movie or a book, consist of one predefined sequential trail, in hypermedia the user interacts with the medium and gains control over a practically endless number of possible trails. He decides where to go and what to perceive next. However, the increased freedom offered by hypermedia entails increased complexity.

7.2. Linear navigation within the trail

In hypermedia systems, there are multiple links from most elements to other elements. When the user navigates to another element, he can again choose from a variety of links associated with it. The sequence of selections leads to a personal trail through hyperspace with all of the previously opened elements embedded in it. The user can go back to previous elements at any point in time and jump back and forth in that trail.

In today's solutions, trails are usually represented as history lists consisting of a linear textual list of the elements visited. It is a one-dimensional collection of the documents' titles. To recognize a previously visited element, the user has to consciously read all the titles. This is a time-consuming and not very intuitive process. In many cases it will fail because too many textual elements without any visual identity are generated in a short time, so that the overview gets lost and the desired element cannot be recognized in the seemingly endless linear list.

7.3. Link classes

Each element in hyperspace is linked to other ones. The most common links are the ones generated by the originator of an element, such as in cross-links to other chapters, explanations of terms, or reference-links to other people's publications. In today's browser tools, an additional link class was established: the related link. Elements that have a similar content are displayed in a linear text list. This class of links is usually not generated by the author but by external systems that analyze elements and index their classified contents in databases. Author-independent link classes offer *additional valuable approaches to navigation* of public space content.

The fact that in today's solutions these links are represented textually causes problems similar to the ones described in the previous section on textual history lists. Known systems do not offer any conceptual model for gaining an overview of multiple "generations" of related elements. As soon as a new element is opened, a new generation of related links is displayed. It replaces the previous one and makes remembering of secondary interesting elements nearly impossible. A visual interface approach to related link classes offering a ro-

bust conceptual model would be the optimum because it could be used most naturally.

The "related" link class is only one of the many that could be imagined. To obtain a more powerful system, the advanced user should have a large collection of different link classes from which he can compile sets of classes suitable for his current approach to the information space. The consequence for the user interface due to the presence of multiple link classes is a higher demand on the representational model that has to handle the link classes, the link-history, and the navigation trail.

7.4. Handling various promising alternatives

Often the user is presented with a quantity of linked elements of which many seem interesting to him. He can either open all of them in parallel or view them sequentially. Opening multiple elements *at the same time* will often be too much for most users to handle due to the massive amount of information confronting them.

Opening the elements *sequentially* represents a cognitive load from another perspective. As soon as the user opens one of the multiple elements, this replaces the current one. Recognizing further promising links from previous elements requires the user to *actively memorize* them (*"which document" and "which position in the document"*). The *cognitive load* of this process exponentially increases when multiple elements have to be remembered.

7.5. Dead ends and recognizing alternative branches from the history

Within a short period of time the number of potentially interesting elements adds up to a large number of alternatives. Each selection implies a process of rating alternatives and decision-making. As the quality of an object can only be completely evaluated when it is opened, in many cases the user finds that he has reached a dead end that does not contain the information he is seeking. In the easiest case, he would go back one step and choose an alternative link from that element to find a more promising trail or content. It might also be that the user chose the "wrong" branch many steps ago. In today's system, he would have to go back through all the steps (or identify the corresponding element by using the history) to then look for a more promising trail starting from that element. This process is time-consuming. The user is confronted with previous contents (which he already rejected because they did not meet his needs) only to find an alternative trail he had already identified at an earlier point in time in his browsing history. Going through all these irrelevant elements is a very distracting task, and in many cases the user will not find the desired element from which the other trail he has been looking in originates.

8. The link manager

The link manager module consists of

- a context space to the currently active element consisting of document internal links, and
- a large, configurable variety of link classes containing links related to the current element,
- a visual log of previously displayed context space generations,
- a visual log of the user's navigation history in the form of a trail linking previously visited elements.

It is a module that externalizes element, link, and trail information in an appropriate representational model and permanently visualizes it. These features are provided by the link manager's *trail* and link section.

8.1. Two sections

8.1.1. The trail section

The trail section serves as a visual history log. It passively records the trails the user generates by navigating in the other system modules (query, space, flow). This module's section becomes interesting for the user when he is looking for a previously displayed element which the other modules no longer display.

8.1.2. The link section

The link section can be used as a dynamic, inspiring basis for navigation. The user is offered a context space for the active element containing links of multiple classes. He selects one of them, perceives the corresponding content, and in parallel is presented with the next "generation" of elements linked from (or to) it. This process can be repeated. With each iteration, the user can choose an element from the newly proposed "current generation" or from previously displayed ones.

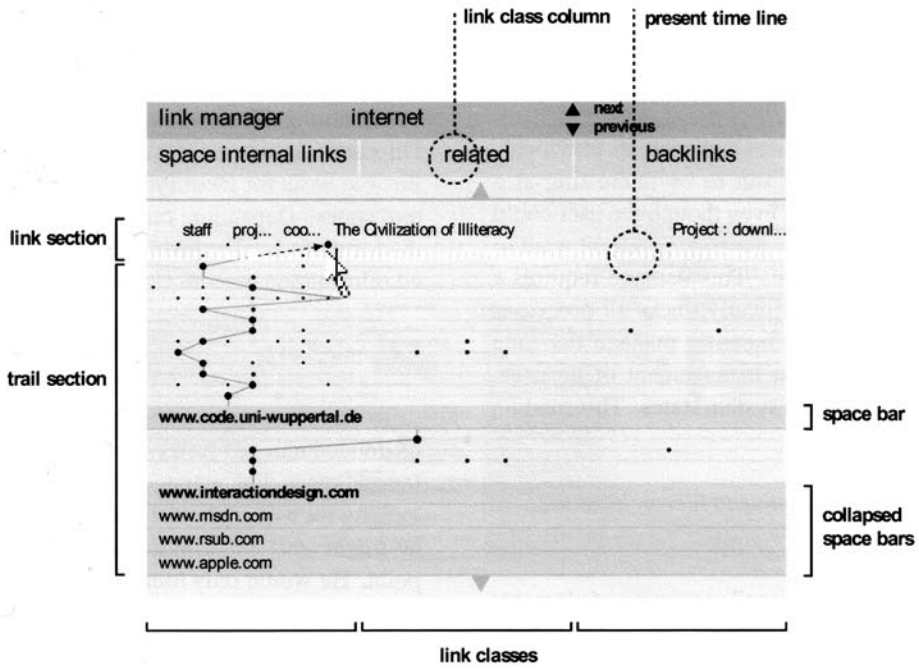

Fig. 7. The components of the link manager interface.

8.2. *The link manager's conceptual model*

The link manager's conceptual model is based on a two-dimensional cognitive map with the time dimension on one axis and the linked elements – divided into different link classes – on the other axis. The user does not have to actively memorize any element because all of them are visually represented. No "endless" history text lists as in today's solutions need to be read. Only concrete positions on cognitive maps in the representational model have to be recognized.

When the user is presented with alternatives of linked elements, he skims them and memorizes the spatial location of interesting elements. When he later returns in order to follow another trail, he can easily recognize the desired element.

The cognitive map has a characteristic visual appearance with multiple structural cues:

- the spatial position of an element,
- different columns for the enabled link classes,
- varied numbers of elements in each row and column of the timeline,
- the navigation trail connecting previously activated elements,
- marked elements,
- subdivisions naming the used module and its content description (e.g., query: "car AND tree," space: "www.interactiondesign.de").

Furthermore, there are content, query, and space elements that differ in their appearance (e.g., color, thumbnail) (see Fig. 7).

8.3. *The link manager interface*

The link manager is a time-based module. The oldest generation of links is in the bottom area – the *trail section*. The second row from the top represents the present time indicated by a white dotted line. The row above is the currently proposed context space consisting of linked elements that are located in the enabled *link classes columns*. Together they form the link section.

The row below contains the previous generation of links, which is one click old. The formerly activated elements are connected by a *trail*. The *space* bars separate the link generations of different spaces. They can be collapsed to hide the contained rows. When the room for the most recent link generations is insufficient, older rows are automatically collapsed so that only their space bars are visible. When the trail section, which grows with every click, does not fit into the link manager instance's pane, the oldest ones are moved out of the visible area and a scroll line is enabled.

9. The flow

9.1. Managing the time dimension

In many cases, system states and contents previously regarded as irrelevant turn out to be interesting at a later point of time. That is, even though the user could have rejected an element in the past, he might want to reconsider it at a later time. This demand requires a system module that automatically tracks all processes and contents. It has to efficiently manage the time dimension, allowing robust management of previous operations, workflows, and system states. This module will be called the flow.

9.2. The flow as an alternative to hierarchical file systems

The flow is a "process space" consisting of elements and module states *automatically tracked by the system.* Like any other space it can be filtered or searched by Boolean queries. Hierarchical spaces (called space in this concept) are best suited for elements or module states that are *actively stored by the user's logic.* These two approaches to information describe not redundancy but rather a powerful symbiosis.

Spaces are *object-oriented.* They are structured *hierarchically* and consist of classes containing elements and further classes. Their structure is defined and managed by the user. He will only sort elements into his (hierarchical) spaces if he is convinced of their usefulness at the moment they are displayed on screen, because it requires a certain effort to classify the object, to sort it into a suiting area of a space, and to assign a name (and perhaps meta-tags) to it. In addition, the user might want to prevent his spaces from being cluttered with too many semi-relevant elements.

Flows are *process-oriented.* They are structured *linearly* by the time dimension and classes of (query, space, content, link manager) elements. They are automatically generated by the system, which logs all processes and system states. The user does not have to pay attention to these system activities running in the background. Examples of tracked processes or system states the user might want to resume from a previous workflow include reception of content, refinement of a query, or identification of a related link. For operations of this class, the flow is the optimum module. To re-open an element (for example, in order to refine a previously defined query), it just has to be clicked and opens in its originating module.

9.3. Scalability of the process space

It must be possible to identify processes and contents by skimming, filtering, and querying the process space. Different representational modes will have to offer a flexible basis for identifying former elements and system states. Depending on the approach, this may take the form of visual or textual (title, abstract, ...) oriented information breadth classes of representation.

9.4. Sub-flows

To obtain an expressive structure, the workflow will be divided into sub-flows – one for each system module (query, space, link manager, content). When the user is looking for a query he defined in the past, for example, he might start by skimming the log at a certain time point. He would only look at the query flow. Depending on his approach, a visual or a textual representation of the Boolean searches might be more suitable. The user might also filter or query a sub-flow (analogous to the complete flow).

9.5. Cross-instance trails

While working with the system the user will use different modules either in parallel or sequentially. For example, he might define a query, open several results in the content module, or navigate in one of the results and copy it to his private space. In this case the user would have produced a trail from the query to the content to the space module. This trail would also be represented in the flow module (if enabled in the settings). There would be a line from the recorded query element to the content element to the space element.

Trails are an effective means of recognizing previously visited elements or system states recorded in the flow. In addition to the characteristic appearance of modules, trails connect elements (in the query, space and link manager) or system screenshots and process bars in the flow module (see Fig. 8).

9.6. Screenshot vs. process mode

Multiple instances of each system module can be in use at the same time. There are two different modes in which all of the resulting module-flows can be represented: screenshot and process mode. Depending on the user's approach, one or the other will be more effective.

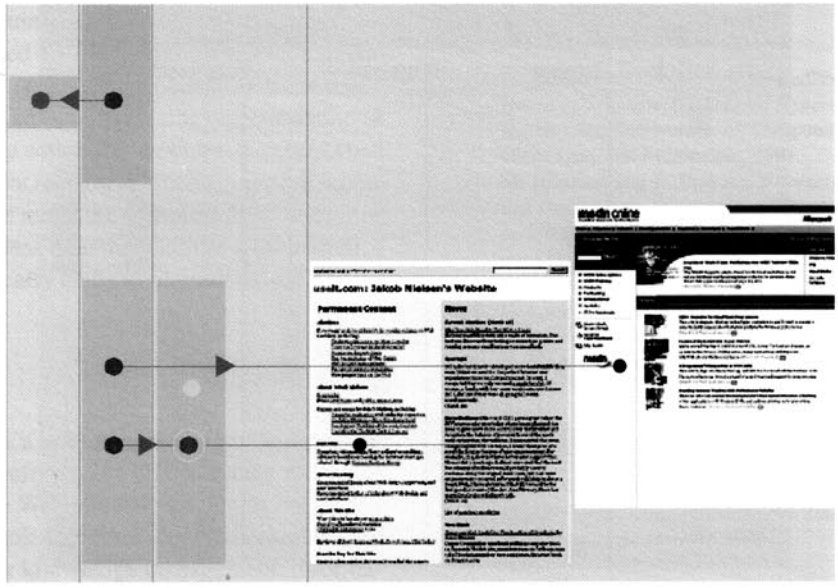

Fig. 8. Extract from the flow module. Elements are connected with cross-instance trails. The three module-flows on the left side are set to minimal information breadth.

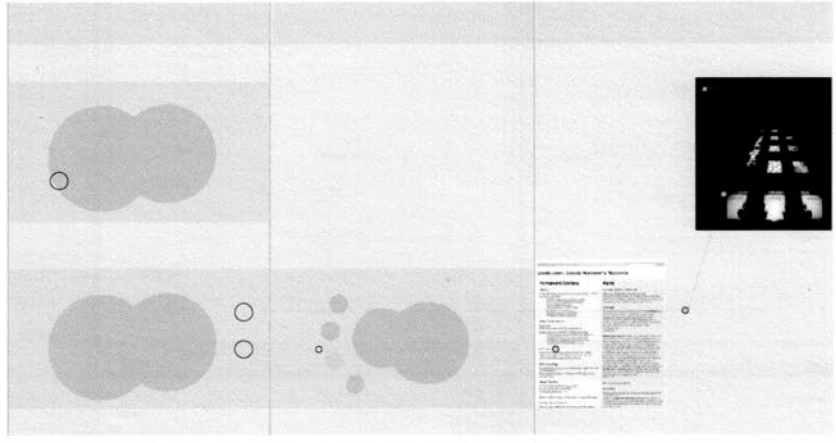

Fig. 9. Flow module in screenshot mode.

9.6.1. Screenshot mode

This mode (see Fig. 9) represents all of the modules' key states by displaying textual (e.g., title, address, query formula) or visual information (screenshots of a visual Boolean query, a content element, etc.) of elements contained in them. This is most effective when the user is searching on an element level and can recognize the concrete element he is looking for. Depending on a module's flow column width, one or more sub-columns containing screenshots can be displayed.

9.6.2. Process mode

This mode is optimized for representing processes. Instead of elements, it represents the duration of processes – active and inactive phases of user interaction within a module. When a module has been operated, a bar is displayed in the corresponding module's column. When multiple instances of a module are opened, they are represented as parallel flows. The representation of module-flows in process mode requires less space than in screenshot mode (see Fig. 10).

It is most effective when the user is searching on a process level and can recognize the combination of modules he used at that corresponding time; a characteristic workflow across multiple modules; or the duration of usage intervals. For example, if the user is searching a query he made some weeks ago, he may

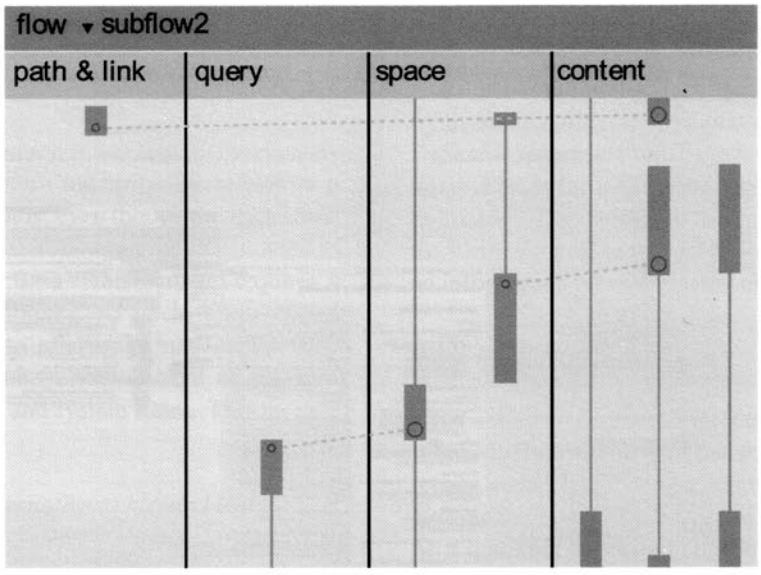

Fig. 10. Flow module in process mode – minimized representation of parallel module instances.

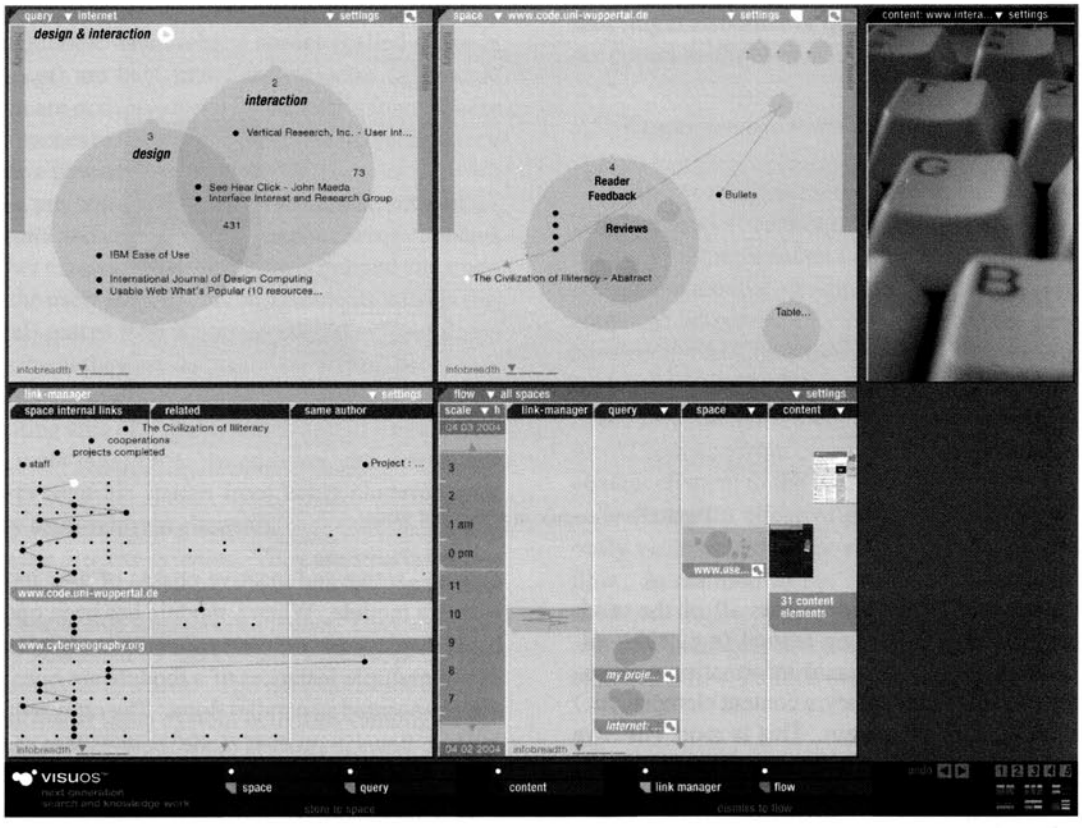

Fig. 11. The visuos operating software with the 4 main modules opened.

have forgotten its visual appearance or exact terms but might remember that he had worked on five queries in parallel or on one very long query followed by two short ones, one of which he opened for a longer time

in the content module. The searched query could then easily be recognized by the visual appearance of the processes and their connection by cross-instance trails.

Analogous to screenshot mode, the representation of cross-instance trails across the modules is a supportive structuring means for recognizing certain system states. When in screenshot mode the complete flow module or a column width is less than the width of a thumbnail, a switch is automatically made to process mode.

Further reading

This article gives a brief overview of the visuos concept. Additional information and interface examples can be found in the Web under www.visuos.com

visuos – the book [3] gives detailed insights into further concepts for knowledge management, information retrieval, research and refinement, monitoring of knowledge resources, and automation and scheduling of tasks.

References

[1] D. Cunliffe, C. Taylor and D. Tudhope, *Query-based Navigation in Semantically Indexed Hypermedia*, Hypermedia Research Unit, Department of Computer Studies, University of Glamorgan, DK Multimedia, 1997.

[2] M. Hertzum and E. Frøkjær, *Browsing and Querying in Online Documentation: A Study of User Interfaces and the Interaction Process*, Department of Computing Science, University of Copenhagen, accepted by ACM Transactions on Human-Computer Interaction, 1995.

[3] C. Lango, *visuos™ – A visuospatial operating software for knowledge work*, Synchron Publishers, Heidelberg, 2003.

[4] R. McAleese, Navigation and Browsing in Hypertext, in: *Hypertext: Theory into Practice*, R. McAleese, ed., Intellect Books, Oxford, 1989.

[5] G.A. Miller, The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information, *The Psychological Review* **63** (1956), 81–97.

[6] Visual Metrics Corporation, *Search Tutorial: Guide to Effective Searching of the Internet*, 1999.

Intelligent Decision Technologies
An International Journal

Volume 2, Number 2, 2008

CONTENTS

Special Issue: Knowledge Visualization

C. Borck
Seeing with the screen 83

J. Hasebrook and A. Saha
Infoviz for strategic decision making 89

C. Lango
visuos™ – Strategies and user interface concepts for next generation knowledge work systems 103

V. Sabol, K. Andrews, W. Kienreich and M. Granitzer
Text mapping: Visualising unstructured, structured, and time-based text collections 117

A. Beer and D. Goldammer
Extreme events 129

K. Neumann
Heart vs. Model 133

Call for papers 139