Semantic Transparency in User Assistance Systems

Andrea Kohlhase
DFKI (German Center for Artificial Intelligence)
Enrique-Schmidt-Str. 5
28359 Bremen, Germany
andrea.kohlhase@dfki.de

Michael Kohlhase
Jacobs University Bremen
Campus Ring 1
28759 Bremen, Germany
m.kohlhase@jacobs-university.de

ABSTRACT

In this paper we analyze the problem of "situating explanations" in user assistance systems. We introduce semantic transparency as a user interface property that enables giving appropriate help. We explicate this notion in document player applications found in office suites, for example. Moreover, we show how semantic transparency can be strengthened when the underlying software is complemented by a semantic ally system. The approach consists in illustrating existing software semantically. We present some semantic extensions of office applications as examples. We also describe how the semantic transparency approach allows the exploitation of new interactions for user assistance systems.

Categories and Subject Descriptors

H.5.2 [**Information Systems**]: Information Interfaces and Presentation—*Interaction styles*; H.5.2 [**Information Systems**]: Information Interfaces and Presentation—*User-centered design*; D.2.2 [**Software**]: Design Tools and Techniques—*User interfaces*

General Terms

Documentation, Design, Human Factors, Theory

Keywords

Semantic Transparency, Interaction, User Assistance, User Interface

1. INTRODUCTION

Modern end-user applications usually come with online manuals, FAQs, on/offline help systems, or discussion forums that are meant to help the user cope with the complexities of interacting with the application [1, 5, 18, 9]. These are jointly comprised under the term "**User Assistance (UA)**" and try to solve two problems: information complexity and situating the explanations. Manuals and FAQs deal with these in an editorial (offline) phase, whereas discussion forums rely on humans to generate appropriate answers.

The case of help systems is interesting since they try to solve the two problems in a semi-automated way. On the one hand, help systems need to cope with huge numbers of information fragments to account for the abundance of potential use problems — a Knowledge Management task; on the other hand, they have to generate explanations at an appropriate level to account for the situatedness of user interaction of individual users — a Human-Computer Interaction task.

In this paper we will analyze the problem of "situating explanations" in UA systems. We point to semantic transparency as a user interface property that enables appropriate help. In Section 2 we introduce the concept in detail and exemplify it using office applications. In Section 3 we will show how semantic transparency can be strengthened in UA systems. In particular, we draw on the Semantic Allies architecture. We will give examples for its usefulness in the form of particular semantic extensions of office applications. Section 4 concludes the paper.

2. SEMANTIC TRANSPARENCY

The field Design of Communication aims at developing and improving communication technologies, and as such is interested in applications, interfaces, *and* documentations. Therefore, it often takes a holistic standpoint and strives for an interdischiplinary approach. This can either be done by a view from without — the "observer/macro-perspective" — or from within — the "**micro-perspective**". Bauer and Gruber showcased in an educational workplace scenario that "*both perspectives have their own explanative power*" [2, p. 676]. They concluded that using the micro-perspective allows to understand to which degree context variables affect use processes. The design of UA systems for any software strongly depends on understanding what triggers users to take up the action of making the software one's own, its "**appropriation**". Thus, in this paper it seems sensible to use the micro-perspective.

Concretely, we ask what the *access points* for appropriation are? The interface objects that carry meaning are natural candidates, because the user acts upon them while interacting with a software application. Note that the interaction is determined by the user's *understanding* and *interpretation* of the interface objects. Thus, we consider the ability of user interface objects to expose the underlying meaning as an important property of user interfaces, as it

SIGDOC '09 Bloomington, Il. USA

directly feeds human-computer interactions.

In Linguistics and Artificial Intelligence (AI) the term "*semantic transparency*" is used. The first field describes it as "*a descriptive phrase that has been used [...] to describe endocentric compounds. Endocentric compound words are those whose whole meaning can be figured out by an analysis of its parts or morphemes*" [24]. Uche Ogbuji described it in a keynote at the Semantic Technology Conference 2005 as "*the ability for machines to properly interpret the content of documents*" [19] representing AI. Meg Houston Maker showed critical interest in the term on her blog post [17]. In particular, she cites Clark with "*intelligence resides at, or close to, the level of deliberative thought. This is [...] the theoretical motivation for the development of semantically transparent systems — ones that directly encode and exploit the kinds of information that a human agent might consciously access when trying to solve a problem.*" [4].

From the micro-perspective and from a Human-Computer Interaction standpoint, we are interested in a mixture, particularly in user interface representations that allow *people* to interpret them properly.

It is well-known that conceptual metaphors may play a central role in appropriation processes. Note that they are not globally valid, we as humans use "*personal metaphors to highlight and make coherent our own pasts, our present activities, and our dreams, hopes, and goals as well*" [16, p. 233]. But as designers we cannot sensibly use a distinct metaphor for every single user interface object.

2.1 Semantic Transparency in User Interfaces

For a precise definition of semantic transparency as we want it, we first need to introduce the notion of 'semantic object'.

For any user interface we assume a reason for its existence — its *intention*. For instance, if we take MS Excel as a user interface for spreadsheets, then its intention is to support ledger sheet functionality. The intention of MS PowerPoint consists in the support of presentations and that of text editors is to help people to create text documents.

The intention determines a set of specific objects that carry its meaning. For example, in Excel we have cells (points in a two-dimensional table), formulas that are assigned to cells, functional blocks (connected cell ranges with the same underlying formula), tables, and legends. In PowerPoint these objects are text boxes, slides, and shows, whereas in Word they are words, sentences, sections, or paragraphs, for instance. These intention-dependent objects we call the **semantic objects** of an application. Note that the semantic objects not only carry meaning by themselves, they may also be related to each other with respect to the intention, for instance legend cells in a spreadsheet may describe a functional block, whereas "smart links" tie together objects in PPT. Observe also that the action/interaction potential of the user interface (UI) objects is largely determined by the semantic objects and their interrelations.

We call a user interface **semantically transparent**, if it enables a user to access its semantic objects and their relations via the corresponding UI objects. If a user interface is semantically transparent, then it allows the user full access to its intention. Note that this is not a question of usability, but of interaction potential. Not every realized interaction is good from the usability standpoint, for instance, Word's notorious automatic capitalizing of words.

This "all-or-nothing" definition severely limits its usefulness. A more fine-granular assessment can be done at the UI object level. We call a UI object **transparent**, if the UI allows access to the relevant information and its underlying semantic relations. Note that we can recast semantic transparency for a user interface now as the property of realizing transparent objects — more semantic transparency translates into better interaction potential.

2.2 Realized Semantic (Non-)Transparency: Office Applications

Perhaps the most simple example of semantic transparency is the "What is this?" feature in the OpenOffice (OO) help system: an interaction mode, where the mouse pointer changes to the one on the right and which displays extended tool tips (help texts) on any interface element the mouse hovers over. It makes the meaning of interface elements transparent to the user.

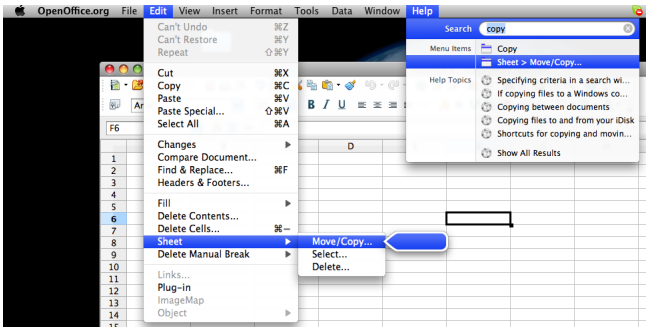

Figure 1: An Apple Help Feature for Menus

The Apple help functionality also offers semantically transparent features. In Figure 1 we see that the help system is used for the term "copy". Here, the user wanted to know more about the where-abouts of the copy-feature, therefore she chose the menu-item "Sheet > Move/Copy...". The help system then provides dynamically the expansion of the corresponding first and second level menu item in the actual UI.

Another form of semantic transparency can be found in office applications that allow the user to manipulate documents via a visual user interface. For instance a spreadsheet program like MS Excel or OO Calc will visualize a spreadsheet document as in Figure 2.

B15 =SUMME(B9:B13)

	A	B	C	D	E	F	G	H
1	**Profit and Loss Statement**							
2								
3	(in Millions)			Actual			Projected	
4		1984	1985	1986	1987	1988	1989	1990
5								
6	**Revenues**	3,865	4,992	5,803	5,441	4,124	4,617	5,223
7								
8	**Expenses**							
9	Salaries	0,285	0,337	0,506	0,617	0,705	0,805	0,919
10	Utilities	0,178	0,303	0,384	0,419	0,551	0,724	0,951
11	Materials	1,004	1,782	2,046	2,273	2,119	1,975	1,84
12	Administration	0,281	0,288	0,315	0,368	0,415	0,468	0,527
13	Other	0,455	0,541	0,674	0,772	0,783	0,794	0,805
14								
15	Total Expenses	2,203	3,251	3,925	4,449	4,573	4,766	5,042
16								
17	**Profit (Loss)**	1,662	1,741	1,878	0,992	-0,449	-0,149	0,181

Figure 2: A Simple Spreadsheet after [25]

Semantically, spreadsheet documents are functional programs represented as arrays of *spreadsheet variables*, which in turn consist of a *current value* and a *formula* (an arithmetic/logic expression involving spreadsheet variable names)

to compute it from the values of other variables [8]. In typical spreadsheet programs, spreadsheets can be directly manipulated via the user interface that uses a grid-like arrangement of cells, rows, and columns inspired by the familiar ledger sheets in accounting [25]. Here, semantic transparency consists in the tight correspondence of grid rectangles with cells. This correspondence is built into the system by naming the spreadsheet variables by a combination of the column and row identifiers of their corresponding cell. The resulting values are directly presented in the grid, and the formulae are displayed in the formula window — possibly using a color coding scheme to tie cell addresses back to cells as a visual aid. Thus, the semantics of the variables can be accessed by clicking on the corresponding cells, and hence, the semantic objects are transparently accessible to the user.

Technically, semantic transparency is directly tied to the way the semantic objects are represented in the underlying software, in particular how the APIs of the data layer give access to them, and how they are mapped to the visual interface primitives. As we have seen, in spreadsheet programs this is very direct, and the spreadsheet programs even export the APIs for spreadsheet variables, cells, rows, and columns for use in third-party extensions of the software.

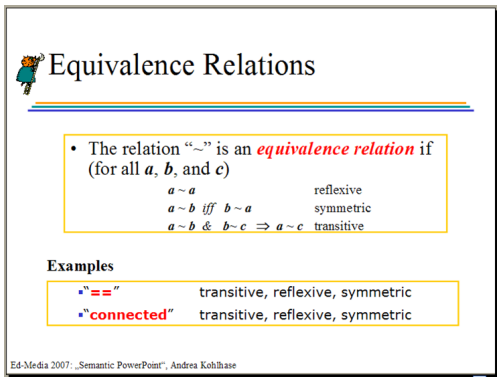

Figure 3: A Simple Presentation Slide

Presentation applications like MS PowerPoint, OO Impress, and Apple Keynote are another class of applications whose interfaces are semantically transparent. At first sight the semantic objects consist in presentation elements like text fragments, itemization, images, and arrows (as seen in Figure 3). They are directly mapped to their obvious visualizations, and the document object model provides classes and methods for their manipulation. For judging semantic transparency in presentation applications though it is crucial to understand that these support several modes with distinct user types in mind (see Section 3.2). For instance, the user interface for animations which is a feature of the slide show mode has a relatively complex and indirect interface in the presentation creation mode, but allows almost no control during slide show mode.

To pinpoint the concept of semantic transparency it is useful to consider a counter-example as well: Word processors like MS Word, OO Writer, or Apple Page. For the discussion below it is critical to note that we distinguish word processors from editors like NOTEPAD or EMACS, which are not specialized for natural language documents but to editing general strings of characters like programs, LaTeX sources, or binary files in hexadecimal form. In contrast text editors, word processors are intended for editing documents in

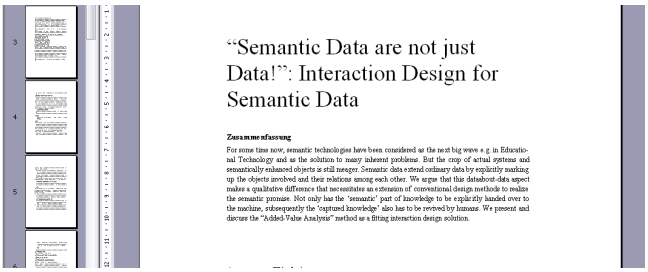

Figure 4: A Simple Text Document

natural language. Thus text words, phrases, and sentences are arguably the meaning-carrying objects — see the document in Figure 4 as an example. Surprisingly, the user interface of the word processors mentioned above has few functionalities for these direct semantic objects. Only the double-click-thus-select feature and the spell-checker come to mind. The supported objects are mainly derived ones like section headings, table of contents, bibliographies, or paragraph styles. Thus, word processors can be viewed as semantically transparent at the document structure level, but rather not at the text level.

The document object model of word processors (basically) represents documents as long sequences of characters with style information, so that the diagnosed lack of semantic transparency can be attributed to the non-semantic representation of their semantic objects. In fact, this can lead to problems. For instance, the capitalization of first words in sentences in MS Word is crippled, since Word does not 'know' the real meaning of a sentence object. Figure 4 shows another example, as here the German heading "Zusammenfassung" introduced by the (semantic) representation of the abstract is out of sync with the English text of the paper: the text language (and its relation to the section headings) is not explicitly represented in the document master.

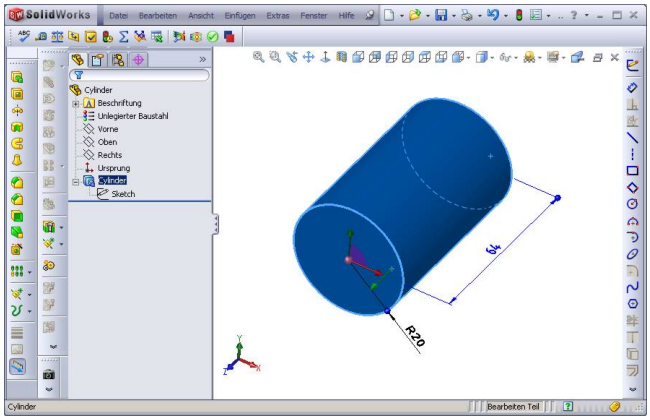

Figure 5: A Simple Construction in SolidWorks

Finally, we want to show that the technical realization of the object-visualization correspondence underlying semantic transparency need not be as simple as in the examples above. We use the SolidWorks CAD/CAM (Computer Aided Design/Manufacturing) system [21] by Dassault Systèmes as an example. In a CAD system, the semantic objects are solid objects and sketches, represented as Boolean combinations of sets of points in two/three-dimensional space. These primitive objects (see Figure 5 for a cylinder constructed as a 2D-disk with a given center and radius extruded along a line)

can be combined into complex *constructions* maintaining geometric constraints on the parameters. The mapping to the visual counterparts involves computation-intensive processes like surface computation, shading, and possibly even ray tracing to give the user a photo-realistic impression of the objects. Note that this mapping has to be reversible for direct manipulation: CAD programs like SolidWorks allow the user to use the mouse to "grab" constructions and turn/tilt/pan them for closer visual inspection, or to push or pull on points or surfaces of the objects for changing them, for example. For this the mapping of the interaction points must be computed back to the semantic level. As the visualization is very direct and the SolidWorks user interface allows to manipulate the objects, we might say that it is a semantically transparent interface. What is missing in all current CAD systems though, is access to complex relations among the semantic objects like width to depth ratios of standard machine screws.

3. ENHANCING SEMANTIC TRANSPARENCY BY SEMANTIC ILLUSTRATION

In this section, we will look at how semantic transparency in user interfaces can be strengthened and in turn can be made use of in user assistance systems. For the former we first present a new architecture for embedding semantic technologies. On the basis of two concrete semantic software extensions we indicate the resulting enhancement of semantic transparency and its exploitation towards new help interactions.

3.1 The Semantic Allies Architecture

Semantic technologies like the *Semantic Web* promise to add novel functionalities to existing information resources adding explicit representations of the underlying objects and their relations and exploiting them for computing new information. The main intended application of the Semantic Web is to combine information from various web resources by identifying concepts and individuals in them and reasoning about them with background ontologies that make statements about these.

It is our experience that in semantically transparent interfaces, the "*points of pain*" (i.e. the points where need for help occurs, see [6]) can be directly mapped to information objects. This enables a novel approach for the use of semantic technologies which we call **Semantic Illustration**: Instead of enhancing resources into semiformal ontologies by annotating them with formal objects that allow reasoning as in the Semantic Web 'paradigm', here a semantic system *illustrates* a software artifact $\mathcal{A}$ (an application, program, or document) with a semiformal ontology by complementing it with enough information to render new semantic services. This approach contains a somewhat analogous requirement phrased in [22]. Conceptually, a help system $\mathcal{H}$ is independent of $\mathcal{A}$, even though an implementation may well integrate it into $\mathcal{A}$. In any case $\mathcal{H}$ is related to $\mathcal{A}$ via an interpretation mapping, so that it can serve as a "**semantic ally**" for $\mathcal{A}$.

Thus the Semantic Illustration approach opens the 'use of semantic data' for non-semantic software applications: Any system with formal data can be mashed up with semantic applications. As long as an application provides access to its semantic objects, their transparency can be enhanced with this architecture. Moreover, if semantic technologies are combined with reasoning about the formal data within the system, they yield new forms of interaction, which we call "**semantic interactions**". Note that semantic interactions are not just interactions based on semantic technology. They are induced by the mash-up of a software artifact and a semantic system.

3.2 Realized Semantic Allies in UA Systems

In Section 2.2 we already pointed at semantic objects and their transparency qualities within office applications. Now, we want to showcase how semantic allies can enhance user assistance by augmenting semantic transparency from within the user interface.

SACHS

An example for the Semantic Illustration approach is our SACHS system [12], a semantic ally for the "**DCS**" — an MS Excel-based financial controlling system in daily use at DFKI (German Center for Artificial Intelligence). In SACHS, we illustrate a spreadsheet with a semiformal ontology of the relevant background knowledge via an interpretation mapping. Then we use the formal parts of the ontology to control the aggregation of help texts (from the informal part of the ontology) about the objects in the spreadsheet. In the following cell references (e.g., [B15]) refer to the spreadsheet in Figure 2.

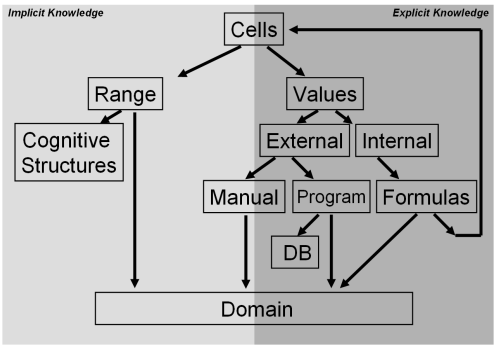

Figure 6: Excel from the Micro-Perspective

We started out with an analysis of DCS usage from a micro-perspective which resulted in the cognitions in Figure 6. In particular, we diagnosed the complexity of understanding Excel documents as an implication of the *incomplete semantic transparency of cells*. On the one hand cells are understood via their values (which are often justified as explicit knowledge), on the other hand they are consumed via their location within a grid structure. For instance, the value in cell [B15] can be interpreted as the result of the formula SUM(B9 : B13) but just as well as "Total Expenses" in the year "1984" using implicit knowledge either about the cognitive structures visualized in grids or simply by ways of domain experience. The latter interpretation is not transparent, for example the understanding of 'total expenses' is only assumed as background knowledge. Interestingly, even the former interpretation is not as transparent as it seems, since formulae typically contain other cell addresses. If for example, domain knowledge about such underlying cells is missing, then the transparency dissolves even though the first step was possibly transparent.

With the SACHS system relevant implicit parts of the domain knowledge become explicit. Importantly, this in-

formation is not only user-accessible just somewhere, but through Excel's usual mechanism for manipulation of cells. Therefore, the semantic transparency of cells is considerably strengthened.

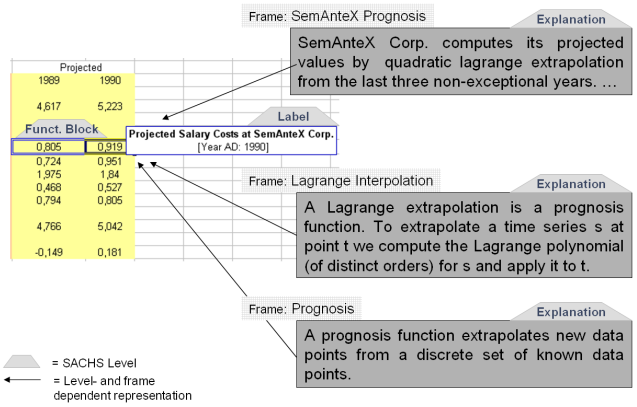

Figure 7: SACHS-generated Help Texts — for Cell [H9] in Fig. 2

Concretely, when a user clicks on a cell, SACHS generates help. She can choose from various help levels. In Figure 7 we can see a (white) SACHS-created *label* for cell [H9]. Additionally, we can catch a glimpse of the different variants of generated help texts on the more detailed *explanation* level for the same cell. These content variants result from a semantic interaction service called "**framing**" (see a detailed description in [14] and a light-weight one in [13]). In a nutshell, SACHS enables the user to choose from a palette of frames for a cell (and thereby a topic) in question. If a user is interested in the meaning of the value in [H9] as a result of a concrete function, then she likes to know what function was applied, whereas if she already knows the name of the function, but doesn't know what it is about, then she likes to see the definition of this concrete function. She might even be wondering, what prognosis functions in general are based on to assess the value of this cell. As framing offers various access points for understanding the meaning of cells, this semantic interaction individualizes semantic transparency.

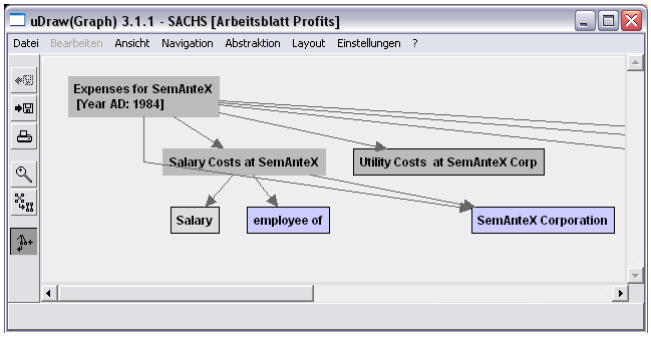

Figure 8: Dependency Graph with 'uses'-Edges — for Cell [B15] in Fig. 2

Another help level consists in the on-the-fly creation of an interactive *dependency graph* when a cell is selected. For example, the graph in Figure 8 is generated, when cell [B15] was clicked. Here, the user has access to all available dependent information for the topic assigned for [B15] in the interpretation mapping. Concretely, all the different expense types like "Salary Costs" and "Utility Costs", that build up the expenses of the company called "SemAnteX", are listed as well as a node for the company itself. A right mouse-click on any graph node opens a content menu that offers to expand the information text in the node itself. Obviously, this is another feature which considerably enhances the (semantic) transparency of cells.

Moreover, we mashed-up the graph-based interface with the interactions needed within a spreadsheet to allow the user to navigate the spreadsheet via the structured background ontology by the definitional structure of the intended functions. In particular, the color-coding of the single nodes indicates whether the concept is connected to a specific cell in the workbook. Darker grey means that it is available on the active spreadsheet, lighter grey hints that the assigned cell is on another spreadsheet but still within the active workbook, and light violet points to a mere semantic concept with no connection to spreadsheets. All grey nodes present *hyperlinks* to respective spreadsheet cells, which all in all results into a **semantic navigation** facility (see [12]). It evolved because the SACHS architecture is based on the Semantic Illustration principle, not because of the use of semantic technology per se.

But SACHS not only enhances the semantic transparency of cells, it also enables access to other semantic objects: **Functional blocks**, which are groups of cells that carry the same intention. The cell ranges [B15:F15], [G9:H9], and [B4:F4] are examples for functional blocks. In particular, in a functional block the cells can be interpreted as input/output pairs of a function. For instance, the cell range [B15:F15] is a functional block, since the cells represent the company's total expenses as a function ϵ of time. Here ϵ is defined as the sum over the corresponding functions of time in [B9:F9], ..., [B13:F13]. Hence, the pair $\langle 1984; 2.203 \rangle$ of values of [B4] and [B15] is one of the pairs of ϵ. The SACHS system offers a functional block mode, in which the cursor highlights the corresponding grid with every click (as showcased in Figure 7). Note that the concept of frames does not depend on cells but really on functional blocks. Therefore, SACHS' interface for framing was implemented in the functional block mode. Another new form of interaction was also realized based on functional blocks as semantic objects: **on-the-fly application of formula variants**. If for instance the underlying formula for cell range [G9:H9] is built up according to the prognosis function "1^{st} order Lagrange interpolation", then a user might be interested in the concrete values resulting from applying a different function like the "2^{nd} order Lagrange interpolation". For a detailed introduction of this semantic interaction we refer to [14] or to a shorter description in [13]. Note that the establishment of the new semantic object 'functional block' allowed the software designer to *discover* innovative interactions by exploiting its (semantic) transparency.

CPoint

Another instance of the Semantic Illustration idea is realized in the CPoint system ("Content for PowerPoint", e.g. [10]), a semantic extension of MS PowerPoint (PPT).

An analysis of PPT use in [11] and of its critical discussion in [7] pointed to three distinct PPT components: The development environment, the slide show, and the document itself. This distinction suggests to discuss user assistance for several user roles within PPT. In particular, there is the "presentation *author*" as a user of the PPT development environment, the "presentation *reader*" as a user of the PPT

document, the "presentation giver/*speaker*" as a user of the PPT slide show, and the "presentation *audience* member" as yet another kind of user of the PPT slide show. Note that this differentiation is not peculiar to PPT, but to presentation software in general.

We argue that distinct user types need distinct semantic objects as the intention of the software changes with the user role taken. The study in [11] implied that MS PowerPoint is optimized for use by presentation authors. Indeed, we have seen in Section 2.2 that the PPT user interface can be considered semantically transparent, if the semantic objects consist of the objects used on slides like the title or the text boxes shown in Figure 3. The semantic transparency breaks down, if one considers the relationship between the text box containing the word "Examples" and the text box describing the examples. The presentation reader or audience member clearly makes out a connection, but the PPT system does not support that at all.

Note that presentation authors tend to switch to the role of presentation reader in order to reuse slides or slide objects created previously. Hence, from the semantic transparency standpoint it is sensible to distinguish a PPT author into a PPT *creator* (someone who creates new PPT objects) and a PPT *aggregator* (someone who reuses already existing PPT objects and thus includes the user role "presentation reader" as well but with a different focus).

In contrast to the PPT interface for a PPT creator, which is semantically transparent, it is not so for a PPT aggregator. Here, the semantic extension CPoint comes into play, since it enables users to enrich PPT objects so that they become transparent for PPT aggregators. Concretely, the CPoint menu bar includes functionalities for categorizing PPT objects, for defining relationships among them, and for conceptual overview and navigation within a collection of PPT documents. For instance, on the slide in Figure 3 the text box containing examples would be categorized as Example and related to the object on the same slide that states the definition of an equivalence relation. The latter would be categorized as Definition and the formal relation between the two objects is characterized as former being an *example for* latter. Note that such categorized objects are new semantic objects, which are especially interesting for PPT aggregators and readers.

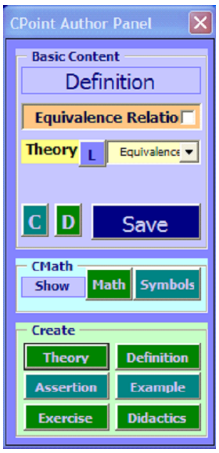

Figure 9: CPoint Author Panel

Now, once this semantic enhancement is in place, CPoint enables a user to grab the intention of a PPT object, for example, by using the author panel in Figure 9. Whenever a PPT object is selected, this panel shows the available basic semantic information. Here the definitional text box was clicked and CPoint shows its category Definition as well as its title and which concept/theory it adheres to.

As mathematical formulae in PPT are mere strings with often 'strange' symbols, their authoring is difficult. Here, CPoint offers to lift them to a semantic object status. As such they become semantically transparent with the CPointAuthor option of switching between their layout and their formula background.

To alleviate the additional burden of semantic annotation for presentation authors, CPointAuthor allows to create pre-categorized objects. In order to enhance the semantic transparency for readers of such objects, their layout is uniformly determined by an underlying local CSS (Cascading Style Sheets) file.

Another semantic object within PowerPoint is the document itself, for example when a series of lectures is given with PPT to build a course. Then each PPT file carries its own meaning for the course. CPoint supports the transparency of this semantic object by offering a theory graph facility. Here, the user can get an overview about used concepts and their relationships. Color-coding is used to express in which file the concept can be found. If for example the definition of the equivalence relation were located in a different file than an example for it, then the author should double-check the situation.

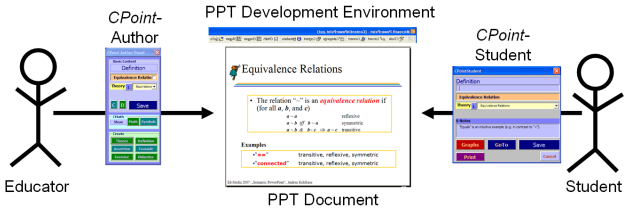

Figure 10: N(=2) Views on 1 Semantic Document

With CPointStudent the CPoint system also offers another view on the same underlying semantic PPT document (see Figure 10). This is geared towards particular presentation readers: students. The basic information is shown in analogy to the CPointAuthor panel functionality, but different services are offered. For instance, it can compile a presentation's content in the form of flashcards (as generated in Figure 11), that can be used by a student for studying for a final exam in the course covered by the according collection of lecture presentations.

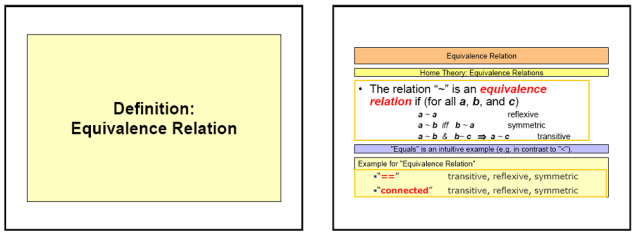

Figure 11: CPoint's Flashcards for Learning

4. CONCLUSION AND FURTHER WORK

We have shown how the concept of semantically transparent interfaces can be utilized to provide fine-granular help systems based on ontology-structured text collections in exemplary applications. We presented the "Semantic Illustration" with the "Semantic Allies" architecture, which is well suited to enhance semantic transparency. Moreover, it enables fine-grained, novel user assistance functionalities.

Interestingly, the presented semantic allies strengthen semantic transparency by exploiting *new* semantic interface objects. Concretely, we saw that SACHS — a semantic extension of MS Excel makes use of functional blocks, whereas

the CPoint — a semantic extension of MS PowerPoint elevates its natural semantic objects to categorized ones. We have demonstrated, that these new semantic objects can be used as anchoring points for (innovative) help interactions like SACHS' content choice by framing or CPoint's flashcard generation. For lack of space we have not discussed the FormalVI system, a semantic ally for the SolidWorks system discussed in Section 2.2, where the geometric objects are enhanced by formal specifications to enable verification of safety properties of the end products. See [15] for a discussion.

Another interesting aspect of semantic transparency is that it supports the notion of 'user as designer' which proved to be so very successful in the Web 2.0 era. In particular, if a semantic object is designed to inform the user of the designer's understanding, then she is enabled to perceive all of the software's (explicit and implicit) affordances and can appropriate it accordingly.

So far, we have considered semantic transparency as an intrinsic user interface property. But having studied the property in some detail, we see the possibility that it is not intrinsic, but dependent on the user's perception. As a consequence, semantic transparency could be integrated into multi-layered interfaces [20]. We leave this for further work.

It is crucial to note that semantic transparency is not just a usability feature, or indeed automatically leads to better user interfaces per se. But a focus on this property allows to exploit new potential interactions that can be especially useful in user assistance systems, since it is at the intersection of Human-Computer Interaction and Knowledge Management. As such it should also be studied further within Communication Design:

"We become the objects we look upon but they become what we make of them" [23, p. 46].

5. REFERENCES

[1] O. D. Andrade and D. G. Novick. Expressing help at appropriate levels. In *SIGDOC'08 Conference Proceedings*, pages 125–130. ACM, 2008.

[2] J. Bauer and H. Gruber. Workplace changes and workplace learning: Advantages of an educational micro perspective. *International Journal of Lifelong Education*, pages 675–688, 2007.

[3] J. Carette, L. Dixon, C. Sacerdoti Coen, and S. M. Watt, editors. *MKM/Calculemus 2009 Proceedings*, number 5625 in LNAI. Springer Verlag, 2009.

[4] A. Clark. *Mindware: An Introduction to the Philosophy of Cognitive Science*. Oxford University Press, USA, 2000. ISBN-13=978-0195138573.

[5] M. Corbin. Design checklists for online help. Online publication (`http://www.writersua.com/articles/checklist/index.html`), Nov. 2004. Accessed on 2009-05-15.

[6] M. Ellison. Embedded user assistance: The future for software help? *interactions*, 14(1):30–31, 2007.

[7] D. K. Farkas. Toward a better understanding of PowerPoint deck design. *Information Design Journal + Document Design*, 14(2):162–171, 2006.

[8] S. L. P. Jones, A. Blackwell, and M. M. Burnett. User-centered approach to functions in Excel. In *ACM International Conference on Functional Programming*, pages 165–176, 2003.

[9] G. Kearsley. *Online help systems: Design and implementation*. Intellect Press, Ablex, 1988.

[10] A. Kohlhase. Semantic PowerPoint: Content and semantic technology for educational added-value services in MS PowerPoint. In C. Montgomerie and J. Seale, editors, *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2007*, pages 3576–3583, Vancouver, Canada, June 2007. AACE.

[11] A. Kohlhase. MS PowerPoint use from a micro-perspective. In *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2008 (ED-MEDIA'08)*, pages 1279–1286, Vienna, Austria, June 2008. AACE.

[12] A. Kohlhase and M. Kohlhase. Compensating the computational bias of spreadsheets with MKM techniques. In Carette et al. [3], pages 357–372.

[13] A. Kohlhase and M. Kohlhase. Modeling task experience in user assistance systems. In *Proceedings of the 27th annual ACM international conference on Design of communication*. ACM Special Interest Group for Design of Communication, ACM Press, 2009.

[14] A. Kohlhase and M. Kohlhase. Spreadsheet interaction with frames: Exploring a mathematical practice. In Carette et al. [3], pages 341–256.

[15] M. Kohlhase, J. Lemburg, L. Schröder, and E. Schulz. Formal management of CAD/CAM processes. In *16th International Symposium on Formal Methods (FM 2009*, LNCS. Springer Verlag, 2009. in press.

[16] G. Lakoff and M. Johnson. *Metaphors We Live By*. The University of Chicago Press, 2003 (1999).

[17] M. H. Maker. Semantic transparency redux. Available at `http://www.engagingexperience.com/2006/02/semantic_transp_1.html`, 2006. Accessed on 2009-07-14.

[18] D. G. Novick and K. Ward. What users say they want in documentation. In *SIGDOC'06 Conference Proceedings*, pages 84–91. ACM, 2006.

[19] U. Ogbuji. XML design for semantic transparency. Online announcement (`http://www.wilshireconferences.com/STC05/AGENDA/A19.htm`), 2005. Accessed on 2009-06-30.

[20] B. Shneiderman. Promoting universal usibility with multi-layer interface design. In *CUU '03*, pages 1–8. ACM, 2003.

[21] *Introducing SolidWorks*. SolidWorks Corporation, Concord, MA, 2002.

[22] T. Tague. The big picture - how semantic technologies introduce a new paradigm for interaction. Invited talk at the Semantic Technology Conference, 2009.

[23] S. Turkle. *Life on the Screen: Identity in the Age of the Internet*. Touchstone, 1997.

[24] Wikipedia. Transparency (linguistic). Available at `www.wikipedia.org`, 2009. Accessed on 2009-07-14.

[25] T. Winograd. The spreadsheet. In T. Winograd, J. Bennett, L. de Young, and B. Hartfield, editors, *Bringing Design to Software*, pages 228–231. Addison-Wesley, 1996 (2006).