CPoint: Dissolving the Author's Dilemma

Andrea Kohlhase[1], Michael Kohlhase[2]

[1]School of Computer Science, Carnegie Mellon University ako@cs.cmu.edu
[2]School of Engineering and Science, International University Bremen
m.kohlhase@iu-bremen.de

Abstract. Automated knowledge management techniques critically depend on the availability of semantically enhanced documents which are hard to come by in practice. Starting from a detailed look at the motivations of users to produce semantic data, we argue that the authoring problem experienced by MKM is actually an *author's dilemma*. An analysis of the content authoring process suggests that the dilemma can partially be overcome by providing authoring tools like *invasive editors* aimed specifically at supporting the content creator. We present the **C**Point application, a semantic, invasive editor for Microsoft PowerPoint, geared towards the OMDoc MKM format.

1 Introduction

Knowledge management technologies are concerned with recovering the content and semantics from documents and exploiting it for automation with an emphasis on web-based and distributed access to the knowledge. The interest in applying knowledge management tools to mathematics (Mathematical Knowledge Management, MKM) is based on the fact that mathematics is a very well-structured and well-conceptualized subject, which makes the knowledge management task simpler and more effective.

Currently, the field of MKM focuses on representation formats for mathematical knowledge (MathML [ABC+03], OpenMath [Cap03], or OMDoc [Koh00], etc.), mathematical content management systems [FK00,ABC+02,APCS01], as well as publication and education systems for mathematics [CNXa,MBG+03].

While the system prototypes showcase the potential *added value* of explicitly representing the content and semantics of mathematical knowledge, they have failed to take off in terms of actual deployments. The main bottleneck seems to be the lack of large-scale, high-quality corpora of mathematical knowledge marked up in the respective MKM formats and the effort involved in creating these. Conventional wisdom (aka. "hope") in MKM is that the added-value applications based on semantic annotations will create a stimulus that will entice common users to invest time and effort into this exciting new technology. But the community experiences at the moment, that it is not yet sufficiently tempting. We will call this the **MKM authoring problem**.

In this paper, we will take a detailed look at the motivations of users to create MKM content and show that the MKM authoring problem is actually

an author's dilemma. We will then develop the concept of an invasive editor as a (partial) solution to the MKM authoring problem and present the **C**POINT application, a semantic, invasive editor for Microsoft PowerPoint.

2 The Author's Dilemma

For a user of MKM material, the *motivation* for preferring semantically rich data is simple: explicit document structure supports enhanced navigation and search, semantic markup yields context and search by content. Furthermore, the higher the degree of semantic structure, the more added-value services can feed on the material, the higher the benefit for the user.

2.1 What is the Author's Dilemma?

For a document author, the *cost* of creating a document is a priori proportional to the depth of the markup involved (assuming that the markup quality is cost-independent in an ideal world). However, once the markup quality passes a certain threshold which supports flexible reuse of fragments, document creation costs may actually go down as they are dominated by the cost of finding suitable (already existent) knowledge elements. Thus, the author is interested in a high reuse ratio, provided that retrieval costs are not prohibitive. The benefits are obvious for the author who has the opportunity to reuse her own content modules frequently, but the *real payoff* comes when she is part of a group of individuals that share content objects and knowledge structures freely. But why should an author share her content modules with others, who could make use of them without contributing to the common share of materials?

Cooperation is often analyzed by means of a non-zero-sum game called the **Prisoner's Dilemma** (see [Axe84]). The two players in the game can choose between two moves, either "cooperate" or "defect". The idea is that each player gains when both cooperate, but if only one of them cooperates, the other one, who defects, will gain more. If both defect both lose, but not as much as the 'cheated' cooperator whose cooperation is not returned. The prisoner's dilemma is meant to study short term decision-making where the actors do not have any *specific* expectations about future interactions or collaborations.

The analogy to the document author's situation is apparent: if the author decides to invest his time and effort and others contribute as well, everyone profits tremendously from this synergy of cooperation. On the other hand, if just the author works on semantic markup, then he will gain nothing in the short run (but some in the long run). So, we can rightfully call it the **author's dilemma**.

In the prisoner's dilemma, if the decision-makers were purely rational, they would never cooperate as they should make the decision which is best for them individually. Suppose the other one would defect, then it is rational to defect yourself: you won't gain much, but if you do not defect you will have all the work. Suppose the other one would cooperate, then you will gain (especially in the long

run) whatever you decide, but you will gain more if you do not cooperate (as you don't have to invest your time and effort), so here too the rational choice is to defect. The problem is that if all actors are rational, all will decide to defect, and none of them will gain anything. If we assume MKM authors to be rational, then we anticipate their non-cooperation. *The MKM authoring problem is a consequence of the author's dilemma.*

In order to tackle the author's dilemma we investigate the central assumption of the prisoner's dilemma that the actors do *not* have "specific expectations about future interactions or collaborations".

One way to get around the author's dilemma is to build or strengthen these expectations, for example by establishing targeted, cooperating research groups, open source and open content licensing, or citation indexes. Such measures[1] may well tip the scale towards cooperation and would therefore be a very worthwhile contribution to the MKM authoring problem.

In this paper, we will single out the 'real payoff' benefit, show why this argument doesn't constitute a specific expectation in the dilemma scenario, and finally dissolve the author's dilemma by changing its input parameters (costs and benefits). In particular, we will concentrate on a single author and the content authoring process.

2.2 The Content Authoring Process

The key intuition in MKM formats is to make semantic structures that are implicit in conventional forms of communication so explicit that they can be manipulated by machines. This explication can happen on several levels: formats like MathML [ABC+03] or OpenMath [Cap03] allow to mark up the structure of mathematical formulae, formats like CNXML [CNXb] mark up the document structure and allow to classify text fragments in terms of mathematical forms like "Definition", "Theorem", "Proof", etc. Formats like the logic-based Casl [CoF98] or the development graph format [Hut00] allow to structure mathematical knowledge into graphs of so-called "Theories", which make semantic relations (like reuse of content, semantic (in)dependence, and interpretability) explicit and available for computer-supported management. Finally, formats like OMDoc [Koh00] attempt to combine all of these aspects of MKM into one integrated representation format for mathematical knowledge.

The explication of structure allows for a new authoring process, the **content authoring process**. In conventional document formats like LaTeX, the document- and knowledge structure is inseparable from the *atomic content* (paragraphs of text, mathematical statements, ...). Therefore they have to be authored at the same time, with the exception of copy-and-paste techniques that have become available by electronic document formats. With the advent of semantic markup techniques, hypertext, and the distribution medium of the Internet, users are enabled to aggregate knowledge fragments (self-authored or from other sources) without losing the semantic context, or having to adapt notations:

[1] See the Connexions project for community-building efforts in this direction [HBK03].

the more structured the knowledge fragments, the simpler their aggregation and reuse.

In fact, approaches like "Learning Objects" [Hod03] postulate that knowledge should be represented in small units like atomic content and that all documents that can be formed using these should be represented as aggregates that mark up the structure and only reference the content. Such aggregates are lightweight, ephemeral structures that can be created on the fly and for special situations, while the learning objects form the heavyweight base of knowledge, that is expensive to create but can be reused in multiple aggregates.

Basically the same idea was realized with the semantic data format OMDoc where a document is considered to consist of a narrative document with links to a content base that is another OMDoc document (see [Koh04c]). Here, the 'learning object' is extended to a 'content object' by taking the context into consideration, enhancing it's semantic value and making it more reusable in the process.

2.3 Dissolving the Author's Dilemma: Creator Support

In the framework of the (new) content authoring process, we can see that the role of a document author, which used to intimately link the activities of content creation, content aggregation, and content presentation can be split up into two roles: **creator** and **aggregator**. The aggregator collects and presents atomic content, whereas the creator builds atomic content so that we might use the term **content author** as well. In fact, we expect that over time the classic academic roles of teacher, survey paper author, or science journalist will be concerned mainly with content aggregation, adding only descriptive text to existing content objects. Currently, the value of semantic markup shows itself only later in added-value services — not at the time of the actual content creation. Not surprisingly, the added-value applications have concentrated on the consumers of the semantic data and not on its producers.

If we reformulate the author's dilemma as a cost-benefit equation in these terms, it reads *"The creator's loss is the aggregator's profit."*. In other words, the allegedly specific 'real payoff' expectation in the author's dilemma scenario really isn't one for the creator of the semantic data. The obvious conclusion consists in separating the single cost-benefit equation into one for the content author and one for the aggregating author. The author's dilemma dissolves if we can give the creator his own motivation for creating semantic data. In particular, **equipping *the creator* not only with specific content authoring support but also with added-value services will help solve the MKM authoring problem**.

Our approach to promote the creator starts with the *concept of an invasive editor*. Generally, authors select document creation systems (editors) with a particular purpose in mind — for instance in the face of presentational tasks. They frequently opt for editors optimized for the intended output media like data projectors (e.g. PowerPoint as presentation editor) or journals (e.g. emacs

as a LaTeX-editor). As these editors optimize their output facilities to the presentational task at hand, they usually don't contain infrastructure for content markup. If an author wishes to annotate his knowledge, then he often has to leave his accustomed editor to use other tools, which is perceived as painful by experienced users. This is clearly a hurdle for an author's willingness to do semantic markup and can be alleviated by providing semantic editing facilities within the chosen editor. We call an editing facility an **invasive editor**, if it is build into an existing application and performs neglected functionalities like content markup. Such an add-on is nurtured by the existing editor in the sense that it adopts its feel, look, and location.

The apparent plus for a content author consists in the fact that she can write *and* markup documents in her usual work environment at the same time. But the real benefit is the following: she can improve the quality of her document by executing content checks and by visualizing the properties (e.g. stringency) of its content. In short, *an invasive editor provides the potential point of entry for creator support and creator-specific added-value services* in her chosen editor.

The concept of an invasive editor is latent in many modern document- or program development environments, and in fact a desired phenomenon. 'Host systems' provide scripting facilities that can be used to create interfaces to other dimensions of information in the documents. For instance, the various office suites offer some variant of VBA (Visual Basic for Applications), formatting systems like LaTeX have a built-in macro processor, and computer algebra systems have embedded programming facilities. A crucial prerequisite for the suitability of a host system consists in the extensibility of the document storage format. We do not consider simple extensions like `emacs` modes for programming languages or semantic data formats as invasive editors, since they only provide editing facilities for dedicated file formats, and lack the added dimension.

In the Course Capsules Project (CCaps [CCa,KSJ+02]) at Carnegie Mellon University we have experimented with two invasive editors with complementary characteristics: nb2omdoc [Sut04], an editor plug-in for Mathematica as a computer-algebra-oriented and therefore mathematically interactive system, and the **C**Point system for MS PowerPoint (PPT) as a purely presentation-oriented editor. We will present the latter in the next section.

3 CPoint – an Invasive Editor for Content in PowerPoint

Before we present the **C**Point application let us consider the discrepancy between existing knowledge and its presentation in PPT. Based on this analysis, we will take a look at **C**Point's forms for semantic data input, its support for content authoring, and finally demonstrate the PPT author's added-value utilities implemented by **C**Point.

3.1 Presentation vs. Knowledge

CPoint's goal is to provide an author with an interface to explicitly store semantic information (knowledge) in the PPT slide show itself without destroying the

presentational aspects of the PPT document. Critical to the task is the apparent gap between the content in a PPT document and the intended communication of knowledge in a PPT talk.

> Knowledge is the psychological result of perception and learning and reasoning
> `http://www.cogsci.princeton.edu/cgi-bin/webwn`

A Priori Content MS PowerPoint is a visual editor and player for slides in a presentation. Thus, it exclusively addresses presentational issues — the placement of text, symbols, and images on the screen, carefully sequenced and possibly animated or even embellished by sound. Obviously, the text and the pictures carry content, and so does the structural, textual, and presentational pattern; we will call it the **a priori content**. In order to assess the quality of a priori content, we list a few typical examples: grouping information in a numbered list implies ranking information, the act of grouping text bubbles in one slide expresses a correlation, or marking text as title with presentational means classifies it accordingly. The superficial and somewhat blurred nature of the a priori PPT content is obvious, as a consequence, the knowledge that is implicit in a PPT presentation cannot be exploited for added-value services.

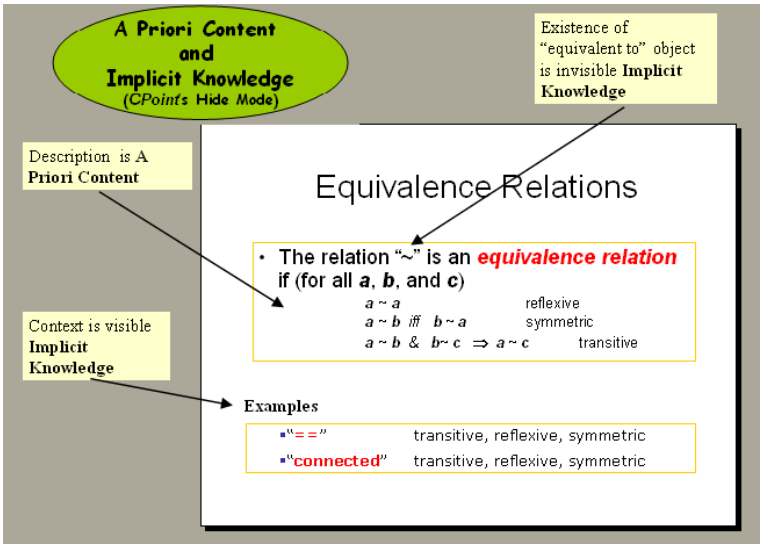


Fig. 1. A Priori Content and Implicit Knowledge in a PPT Presentation

Implicit Knowledge The audience in a PPT talk perceives the a priori content in the PPT document together with what the lecturer says. This is followed by the user's learning and reasoning process: content becomes knowledge by categorizing the perceived information and combining it with what a user already

knows. The author on the other hand already has this knowledge and while he is creating the PPT document he is using it implicitly. So the 'real' content is hidden beneath the presentation form and has to be captured and stored to make it available to MKM techniques. Figure 1 shows the interplay of a-priori content and implicit knowledge. The global context, e.g. the placement of one lecture in an entire course, is another part of the implicit knowledge. Following OMDoc, CPoint captures this via the notion of a **collection**, encompassing a group of inter-related PPT presentations.

Explicit Knowledge CPoint provides functionality to make the implicit knowledge in a PPT presentation explicit. The PPT content author is supported in

- marking up the ontological role of a PPT object
- annotating its relation to the local and global context, i.e. to the just-learned, about-to-be-learned, and assumed knowledge elements, and
- adding background knowledge that is presupposed in the course but not presented in the slides.

We will call the resulting, explicitly annotated knowledge PPT **content** in the following. The two-stage annotation process will be described in detail in Section 3.2. For the transition from implicit to explicit knowledge see Figure 2.

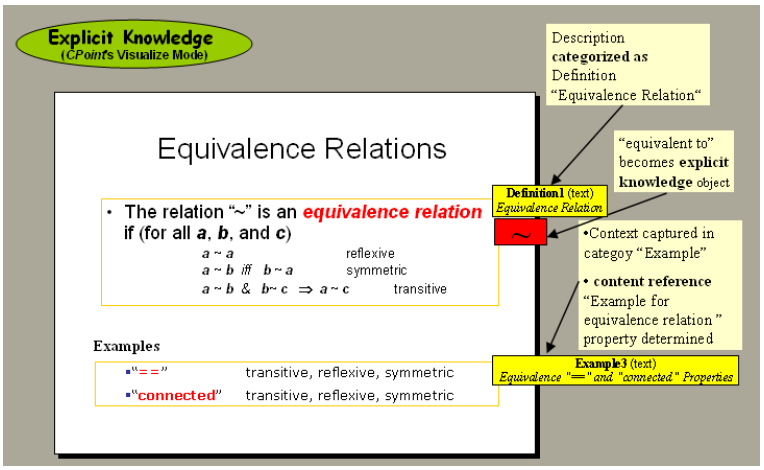


Fig. 2. Explicit Knowledge in a PPT Presentation

3.2 The CPoint Application

CPoint [Koh04a] is a PPT add-in that is written in VBA. To store the semantic information in the PPT files, it makes use of the fact that PPT objects can be

persistently 'tagged' with arbitrary strings. **C**POINT is distributed under the Gnu Lesser General Public License (LGPL [FSF99]), the newest version can be downloaded from `http://cs.cmu.edu/~ccaps`.

Fig. 3. The **C**POINT Menu Bar

The **C**POINT add-in makes its functionality available through a toolbar in the PPT menu (see Figure 3) where it is at an author's disposal whenever the PPT editor is running. The top-level structure of a PowerPoint presentation is given by slides. Each slide contains PPT objects, e.g. text boxes, shapes, images, or tables. By using **C**POINT the author can attach additional information to each PPT object so that it becomes a **semantic object**.

As **C**POINT wants to model the implicit knowledge in a PPT presentation and aims at facilitating the annotation process, it is geared towards the understanding process. Therefore we will continue with the application's illustration along the process' characteristics: categorizing and combining.

Categorizing: The CPoint Categorize Form The very first step in the categorizing process of an object is a naming act (title assignment) which lifts its content from e.g. mere text to a knowledge object (see Figure 4).

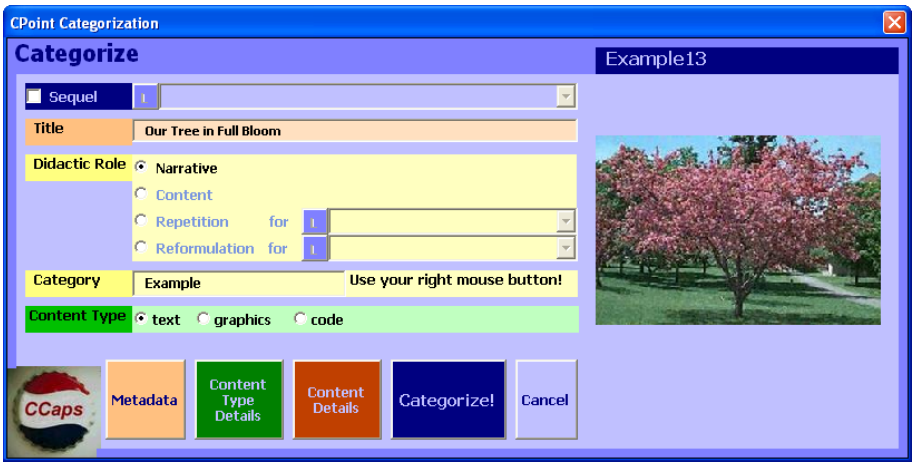


Fig. 4. The **C**POINT Categorize Form

Classification is neither simple nor unique. First, individual styles vary a lot. Secondly, objects like texts or images may be used in different ways: An image

of a tree in full bloom for instance is used narratively in a lecture about trees in computer science whereas a textual definition of a tree in the same lecture clearly contains knowledge. On the other hand the tree's picture may be definitional in a lecture about blossoms in biological science. Furthermore, objects can be pure repetitions and even though they might contain content, not all appearances are used as content objects. Analogously, a knowledge element might be described more than once in a presentation, so that the object and its reformulations are equivalent. Therefore **C**POINT distinguishes an object's **didactic role** in a PPT presentation to be narrative, content, or that of a repetition or reformulation for another object.

The didactic role restricts the subsequent *category selection* — available as a drop-down menu by a right mouse click — as it is only compatible with a subset of pre-defined categories. Categories range from formal mathematical categories like "Theory", "Definition", or "Assertion" to didactic elements.

Sometimes, components of what should be categorized as a single knowledge element are spread over several slides for presentational reasons, possibly interrupted by excursions here and there. In such cases, the original assumption that (groups of) PPT objects directly correspond to knowledge objects is no longer valid. For such situations, **C**POINT provides **sequel objects** that tie a component to the respective previous knowledge element part. These are not individually categorized, the first in line contains the semantic data for the sequel list.

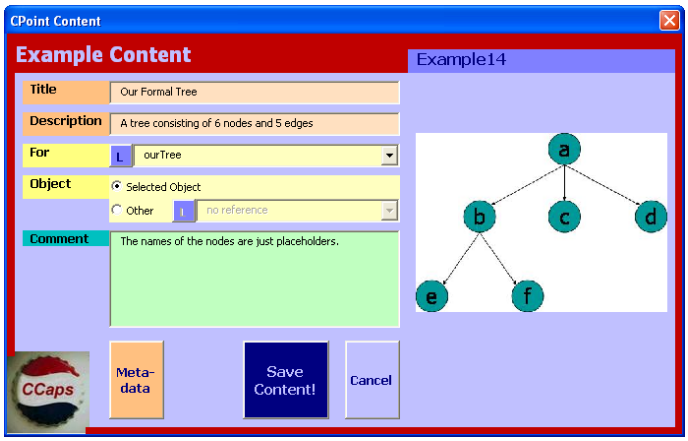


Fig. 5. The **C**POINT Content Form for an Example

It is not always clear what an object's content is (e.g. look at 'the' object in Figure 5 and think of it as ungrouped set). In particular, the presentational information might contain more (category-independent) knowledge than is explicit. Therefore **C**POINT allows to differentiate an object's **content type** to influence the conversion process. The value "text" (the default) results in a di-

rect incorporation of the text value into the external representation. The content type "graphics" triggers the creation of a picture of the object itself and/or all underlying objects at conversion time. This is useful if only the combination of objects describe a knowledge element like a set of circles with single letters and lines between them may illustrate a tree. The content of each object is not noteworthy (e.g. a letter), but their placement in the entity is. Finally, the content type "code" is a specialization for the original application to Computer Science content in the CCAPS project. We anticipate that future applications of **C**POINT will extend the repertoire of content types.

Combining: The CPOINT Content Forms After a PPT object has been classified, we must make its relation to the knowledge context explicit via the respective detailed content form.

In Figure 5 we see the content form for a PPT object. Here, a specific tree (consisting of nodes and edges) is an example *for* the concept of a (formal) tree. In this case, the example is the PPT object itself (witnessed by the selection in the Object field of the form). If the PPT object were e.g. a text box with "Example: a directed graph unrolled" this would serve the purpose of an example (and thus would have been categorized as "example" in the previous step), but the text object itself only serves as description of another object (the directed graph) which *is* the real example and should be referenced in the Object field of the form.

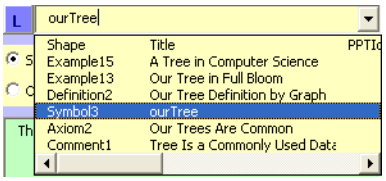
3.3 CPOINT's Support for the Content Author

The following **C**POINT services equip the PPT author with creator-specific tools for content authoring. Many of them directly allow the author to connect her content to a wider knowledge context of MKM materials. The CPOINTAUTHOR **panel** provides a tool palette for displaying and editing central **C**POINT annotations of the currently selected object. While the facilities described in the last section concentrated more on the semantic annotation process for pre-existing text objects, CPOINTAUTHOR focuses on the creation of semantic objects in the content authoring process The presentational properties of these are preset by the authors personal preferences, which can be individually configured in a CSS file (Cascading Style Sheets [Bos98]) associated with CPOINTAUTHOR.

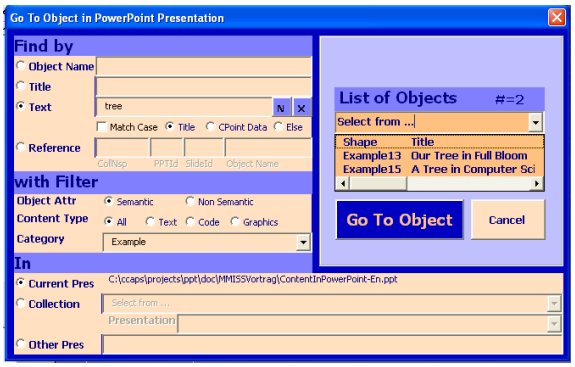
Visualize Mode As the semantic markup must not disrupt the presentational aspects of a PPT document, **C**POINT provides a so-called **visualize mode** at design time. By activating it, annotation labels for semantic objects are created that contain the category information as well as the title and content type of an object. At the same time invisible objects like symbols and abstract objects are visualized. An associated **hide mode** clears the labels.

The Navigator Button Generally, the purpose of a content form is to enter referential information that elucidates an object's contribution to and dependence on the knowledge context. Since such references can be local (inside the current slide show), in the current collection, or even in some external knowledge source (e.g. the MBase mathematical knowledge base), finding and specifying reference targets is one of the major difficulties of semantic markup in general.

For each of the scopes, **C**Point determines the possible target elements (silently loading the necessary documents or connecting to external knowledge sources) and displays them in the adjoining selection box. Since this will normally be too many for a drop-down menu, the user can restrict the search space by various filters (e.g. category) available on right-click. In the figure on the right we can see the Navigator Button and the list of target objects in the **L**ocal presentation.



Navigation As additional navigational help **C**Point offers the **GoTo** interface. The author may search for objects with certain criteria, restrict the found set by filters, determine in what presentation to look for objects, and if he selects one object, he can go to that object directly. On the right we can see that the user searched in the active PPT presentation for all objects which have the word "tree" in their title and are categorized as example. The PPT show contains two yielding objects which are collected in the selection box on the upper right hand of the form.



Export to OMDoc The **convert** functionality allows a user to convert a fully (**C**Point-)edited PPT presentation into a valid OMDoc document from within the PPT editor. Other OMDoc sublanguages are also supported. In particular, it can generate a document in presOMDoc, which is common OMDoc extended by presentational elements, and amOMDoc which can be read by the ActiveMath application.

Note, that the conversion utility recognizes TeXPoint [Nec] inlays, the underlying LaTeX code is conserved (as `code`) in the output OMDoc file.

Import from OMDoc As a PPT document contains a lot of presentational information, **C**Point's import is based on presOMDoc documents. These can be **import**ed from within the PPT editor. The presOMDoc file is first read in, then a new PPT presentation is generated from these parsed OMDoc elements

yielding the presentational information present in the document. The generalization to an import feature of OMDoc documents with the usage of an individual CSS file is at planning stage.

Connection to the Outside World CPoint exports files, which then can be **access**ed by the author directly. Furthermore, the documents are opened with an editor according to their file type and the user's personal preferences. Therefore, the author could read for instance generated OMDoc files with an emacs editor with OMDoc mode.

Editor Notes An editor notes module CPointNotes is available. The author can create (groups of) notes, searching in one or all groups for his notes and jumping from one to another. If an author for instance does want to supply background information, but wishes to finish creating the lecture first, he tags a missing reference by setting an editor note in the group "background" to remind him of inserting the missing references later on. At the same time he finds the phrasing of the text still wanting, so he creates another note for this object in the notes group "polish".

3.4 Added-Value for the Content Author

In the CPointGraphs module [Koh04b] the user is enabled to view the annotated structure in a graph format, i.e. the dependency tree of the knowledge elements is visualized. It offers several distinctive views from a general survey of theories in a collection or presentation to a single detailed theory graph. In Figure 6 we get an idea how extensive the knowledge in a course really is. Note for example, that nodes without dependencies might be considered superfluous.

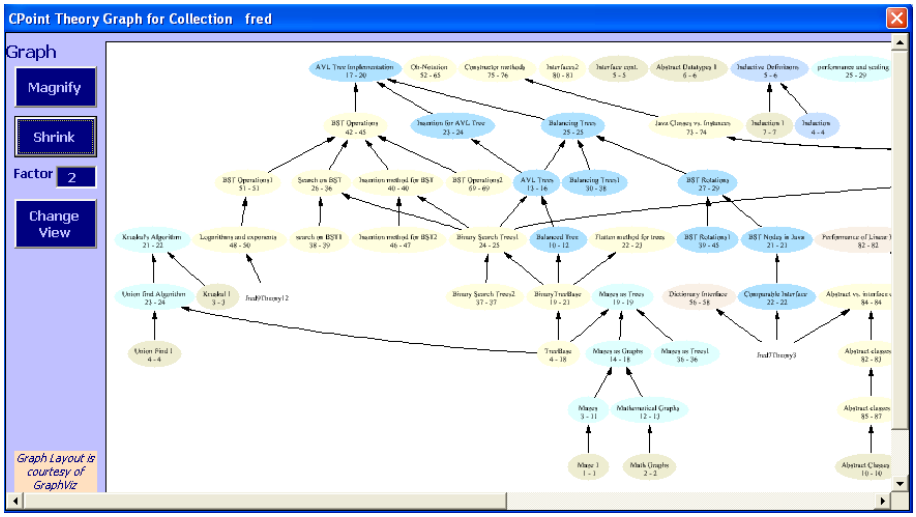


Fig. 6. CPointGraphs: The Theory Graph for a Course

The CPointAM module contains an integrated development environment for ActiveMath content. It allows a user to start and stop the ActiveMath

application, to check for errors and to set parameters. Furthermore, it includes a utility for converting an entire PPT collection into an ActiveMath course. Additionally, it implements an ActiveMath guided tour button in the context menu of each object. This button causes ActiveMath to create an individualized, self-contained document that leads up to the knowledge embodied in this object [MBG+03].

3.5 System Evaluation and Case Study

The **C**Point system has co-evolved with a large case study, the course "15-211 Fundamental Data Structures and Algorithms". 15-211 is an undergraduate course at Carnegie Mellon University (taught twice annually over a period of 10 years by a group of 10 faculty members). The current base of course materials contains about 2500 PPT slides, of which about half have been annotated in **C**Point, the rest being duplicates, variants or obsoleted versions. Our intuitions of the MKM authoring problem and many of **C**Point's auxiliary features have been shaped by this case study. Annotation time per slide (for a talented student) was about 10 minutes, which leads to a 60% MKM overhead, if we assume an average slide creation time of 15 min.

The case study revealed, that without adding background material (that is conveyed orally by the teacher during the lecture) the annotated slide show is too thin as the only resource of content for an MKM system like ActiveMath. The perhaps most surprising result in the case study was that the mental model the authors had of their course materials was changed by the semantic annotation process, resulting in more structured slides, more emphasis on prerequisites, and less presentational gimmicks.

4 Conclusion and Future Work

In this paper, we address one of the central questions for the success of MKM techniques "*Why is it so hard to motivate people to annotate their documents semantically when they agree at the same time to its usefulness and even to its exciting potential?*". We characterize the underlying problem as an author's dilemma and argue, that the alleged pay-off for semantic annotation is not a *specific* expectation for a content author. This predictably leads to the MKM authoring problem. We propose the adoption of invasive editors with creator support and creator-specific added-value services to dissolve the content author's dilemma. We present the **C**Point system, an invasive, semantic editor in Microsoft PowerPoint and illustrate its content author support. We expect that invasive editors will lower barrier for authors, help them manage their extensive collections of presentations more effectively, even without assuming cooperation benefits by sharing materials.

However, the surprising result of research on the prisoner's dilemma is that cooperation spontaneously emerges even in such a hostile situation, if the experiment is iterated and a subset of players experiment with altruism (see [BW02]

for details). Qualitatively, it is safe to say that cooperation emerges earlier, converges sooner, and tolerates more freeloaders, if the cooperation benefit is strengthened and the cost of unreturned cooperation is mitigated. This suggests that invasive editors will also play a role in fostering cooperation.

In the future, we envision a new PPT add-in module CPointPresenter supplementing the existing module CPointAuthor for the different roles a user can play. A presentation author (a presenter) will be supported during the composition of a presentation. He will search, find, and build new presentations based on existing knowledge elements. Here, we want to stress and facilitate the aggregator function of e.g. a lecturer. For a content author we can conceive further support by the application of information retrieval techniques to suggest categories and references for content objects. This will be particularly helpful for the migration of legacy materials to semantically enhanced formats like OMDoc. As knowledge is present in other document formats as well (probably even more so), another goal is the implementation of a **C**Point clone in MS Word. Finally, we plan to connect **C**Point to the MBase system [KF01], so that all knowledge captured in OMDoc documents and stored in an MBase can get directly used.

Acknowledgments The development of the **C**Point system has been funded by the National Science Foundation under grant CCF-0113919 and a grant from Carnegie Mellon's Office for Technology in Education. The authors would like to thank Frederick Eberhardt for his markup of 15-211 and fruitful discussions on the **C**Point system.

References

[ABC+02] Stuart Allen, Mark Bickford, Robert Constable, Richard Eaton, Christoph Kreitz, and Lori Lorigo. FDL: A prototype formal digital library – description and draft reference manual. Technical report, Computer Science, Cornell, 2002. `http://www.cs.cornell.edu/Info/Projects/NuPrl/html/FDLProject/02cucs-fdl.pdf`.

[ABC+03] Ron Ausbrooks, Stephen Buswell, David Carlisle, Stéphane Dalmas, Stan Devitt, Angel Diaz, Max Froumentin, Roger Hunter, Patrick Ion, Michael Kohlhase, Robert Miner, Nico Poppelier, Bruce Smith, Neil Soiffer, Robert Sutor, and Stephen Watt. Mathematical Markup Language (MathML) version 2.0 (second edition). W3c recommendation, World Wide Web Consortium, 2003. Available at http://www.w3.org/TR/MathML2.

[APCS01] Andrea Asperti, Luca Padovani, Claudio Sacerdoti Coen, and Irene Schena. HELM and the semantic math-web. In Richard. J. Boulton and Paul B. Jackson, editors, *Theorem Proving in Higher Order Logics: TPHOLs'01*, volume 2152 of *LNCS*, pages 59–74. Springer, 2001.

[Axe84] Robert Axelrod. *The Evolution of Cooperation.* Basic Books, New York, 1984.

[Bos98] Cascading style sheets, level 2; CSS2 specification. W3c recommendation, World Wide Web Consortium (W3C), 1998. available as `http://www.w3.org/TR/1998/REC-CSS2-19980512`.

[BW02] Andreas Birk and Julie Wiernik. An n-players prisoner's dilemma in a robotic ecosystem. *International Journal of Robotics and Autonomous Systems*, 39:223–233, 2002.

[Cap03] The OpenMath standard, version 2.0. Technical report, 2003. The OpenMath Society, `http://www.openmath.org/standard/om20`.

[CCa] The course capsules project. `http://aiki.ccaps.cs.cmu.edu`.

[CNXa] CONNEXIONS. `http://cnx.rice.edu/`.

[CNXb] Basic CNXML. `http://cnx.rice.edu/content/m9000/latest/`.

[CoF98] Language Design Task Group CoFI. Casl — the CoFI algebraic specification language — summary, version 1.0. Technical report, `http://www.brics.dk/Projects/CoFI`, 1998.

[FK00] Andreas Franke and Michael Kohlhase. System description: MBASE, an open mathematical knowledge base. In David McAllester, editor, *Automated Deduction – CADE-17*, number 1831 in LNAI, pages 455–459. Springer Verlag, 2000.

[FSF99] Free Software Foundation FSF. GNU lesser general public license. Software License available at `http://www.gnu.org/copyleft/lesser.html`, 1999.

[HBK03] Geneva Henry, Richard G. Baraniuk, and Christopher Kelty. The Connexions project: Promoting open sharing of knowledge for education. In *Syllabus, Technology for Higher Education*, 2003.

[Hod03] H. Wayne Hodgins. The future of learning objects, 2003.

[Hut00] Dieter Hutter. Management of change in verification systems. In *Proceedings Automated Software Engineering (ASE-2000)*, pages 23–34. IEEE Press, 2000.

[KF01] Michael Kohlhase and Andreas Franke. MBase: Representing knowledge and context for the integration of mathematical software systems. *Journal of Symbolic Computation; Special Issue on the Integration of Computer algebra and Deduction Systems*, 32(4):365–402, 2001.

[Koh00] Michael Kohlhase. OMDOC: An open markup format for mathematical documents. Seki Report SR-00-02, Fachbereich Informatik, Universität des Saarlandes, 2000. `http://www.mathweb.org/omdoc`.

[Koh04a] Andrea Kohlhase. **C**Point *Documentation*. Carnegie Mellon University, 2004. Technical Manual `http://www.faculty.iu-bremen.de/mkohlhase/kwarc/software/CPoint.html`.

[Koh04b] Andrea Kohlhase. **C**PointGraphs *Documentation*. Carnegie Mellon University, 2004. Technical Manual `http://www.faculty.iu-bremen.de/mkohlhase/kwarc/software/CPointGraphs.html`.

[Koh04c] Michael Kohlhase. OMDOC *An open markup format for mathematical documents (Version 1.2)*. 2004. Manuscript, `http://www.mathweb.org/omdoc/omdoc1.2.ps`.

[KSJ+02] Michael Kohlhase, Klaus Sutner, Peter Jansen, Andrea Kohlhase, Peter Lee, Dana Scott, and Matthew Szudzik. Acquisition of math content in an academic setting. In *Second International Conference on MathML and Technologies for Math on the Web*, Chicago, USA, 2002.

[MBG+03] Erica Melis, Jochen Büdenbender, George Goguadze, Paul Libbrecht, and Carsten Ullrich. Knowledge representation and management in ActiveMath. *Annals of Mathematics and Artificial Intelligence*, 38:47–64, 2003.

[Nec] George Necula. TeXPoint. Program Home Page at `http://raw.cs.berkeley.edu/texpoint/index.htm`.

[Sut04] Klaus Sutner. Converting MATHEMATICA notebooks to OMDoc. to appear in [Koh04c], 2004.