Software Demonstration: **Documents and Knowledge Bases: Engagement Should Lead to Marriage**

Doug SKUCE, Ottawa, Canada

Abstract
We describe how we link knowledge bases to associated web documents. The document provides more information and more pleasant reading, whilethe knowledge base contains highly condensed lexical and encyclopedic (critical) facts. The idea is to be able to move easily by just one click between the two. We have used the Sun Java Tutorial in an undergraduate course on programming languages as an example. We suggest that in the future, web documents could be supplied with such associated knowledge bases to make them more complete and understandable.

1 Introduction: The Problem

Most on-line knowledge is currently stored as web documents. Typically at present these are simply paper translated into electrons. But they leave much to be desired, as we all know. In fact, a fancy new term for such knowledge is "knowledge base", as used, for example, by Microsoft to dignify their unfathomable web site. Seasoned knowledge engineers would not use this term for such an entity. For them, the term "knowledge base" implies order, precision, clarity, and ease of navigation. But how would *you* describe Microsoft's "knowledge base"?

So how might we improve knowledge storage and retrieval beyond merely using computers as *document repositories*? This problem is usually termed *knowledge management (km)*: representing, processing, storing, and retrieving knowledge. But the current technology is clearly still too document-centric. The state of the art in km, at least on 99% of web sites, is HTML or occasionally XML documents on (rarely) a well-organized site, searchable by a good search engine that retrieves lists of whole documents, but not individual facts. It is meant only for patient humans. But often one wants only *facts*, eg "when did FDR die?", or "what is the price of tea in China?" Here the present technology fails miserably: it leaves you with a pile of electronic paper to read, and read, and skip, and read,. . . , and often just give up.

Apart from needing better tools to assist in merely reading, we desire as well tools to assist in document production, creation, maintenance, or even translation. We term all such tools "km assistants". Our lab (www.site.uottawa.ca/lake; LAKE = Language Analysis and Knowledge Engineering) has been working on such tools for about 15 years. Most of these still require some skimming or reading, but a major goal is to minimize this valuable human time. In (Meyer and Skuce 92), we proposed the idea of a *terminological knowledge base (tkb*) that stores detailed lexical *and* domain knowledge in a highly structured database, so one can go "straight to the facts". But in the past, documents and the associated tools and knowledge have been only loosely connected, if at all. We seek to rectify this: we believe that documents should be *intimately connected with supporting tools and knowledge resources. In fact, we are old-fashioned: we want them to be married.*

2 The Solution

Hence we seek to *merge* documents and tkbs into a hybrid we are calling a *dkb* (document/knowledge base), that intimately integrates XML documents with a tkb. Hence one may jump back and forth, going from the document to the tkb for clarification or enlightenment, or the other way for broader or background information. The tkb provides both *lexical* information and as much *encyclopedic* knowledge as desired in the form of crisp, succinct facts.

We have built a prototype dkb that has been used for teaching Java, and the students love it. Its source document is the *Sun Java Tutorial*, a typical web document produced by Java's creator (www.javasoft.com) with the usual warts (eg the most critical concept, class, is given *three different definitions* in different locations. Pity the student studying for an exam.). Our tkb shows this, but identifies one as the desirable one.

The dkb system is called System X (at the moment we have not yet assigned a value to the variable X). It can be used in any situation where technical knowledge must be created, processed and consumed by the users, eg students, translators, or programmers. X merges ideas from databases, AI, IR, and web technology. X's dkb offers all the advantages of web technology, plus it is built on top of a conventional database (we use Oracle, but any dbms will do). Thus it becomes possible to use X in both a *distributed* and *collaborative* manner, which is particularly useful during document manipulation. Facts are entered into the tkb either by selecting them in a document for addition to the tkb, or by creating new ones. If a new fact is added, it will appear in both the tkb and the document, identified by special XML tags. The same of course holds for existing facts that get imported into the tkb. If you have an existing good text to work from, creating the tkb is not hard, provided one has the subject matter knowledge,and hopefully our tools. Now if one is writing a *fresh* document, the tkb construction should go hand-in-hand. We are now doing this for another course.

X is a descendent of our earlier CODE and Ikarus systems (see papers by Skuce). The concepts in the tkb are arranged in precise inheritance hierarchies: if a cat is a mammal and a pet, one may choose to see all the mammal and pet facts listed below those for cat when you select cat in the tkb database. X now has all the features we consider essential: it is web-based (written in Java) and stores its data remotely in a shareable dbms, so speed, scalability, and multi-user access are not a problem. It links easily to any web resource thanks to the dbms.

X has two main parts, not counting the dbms:

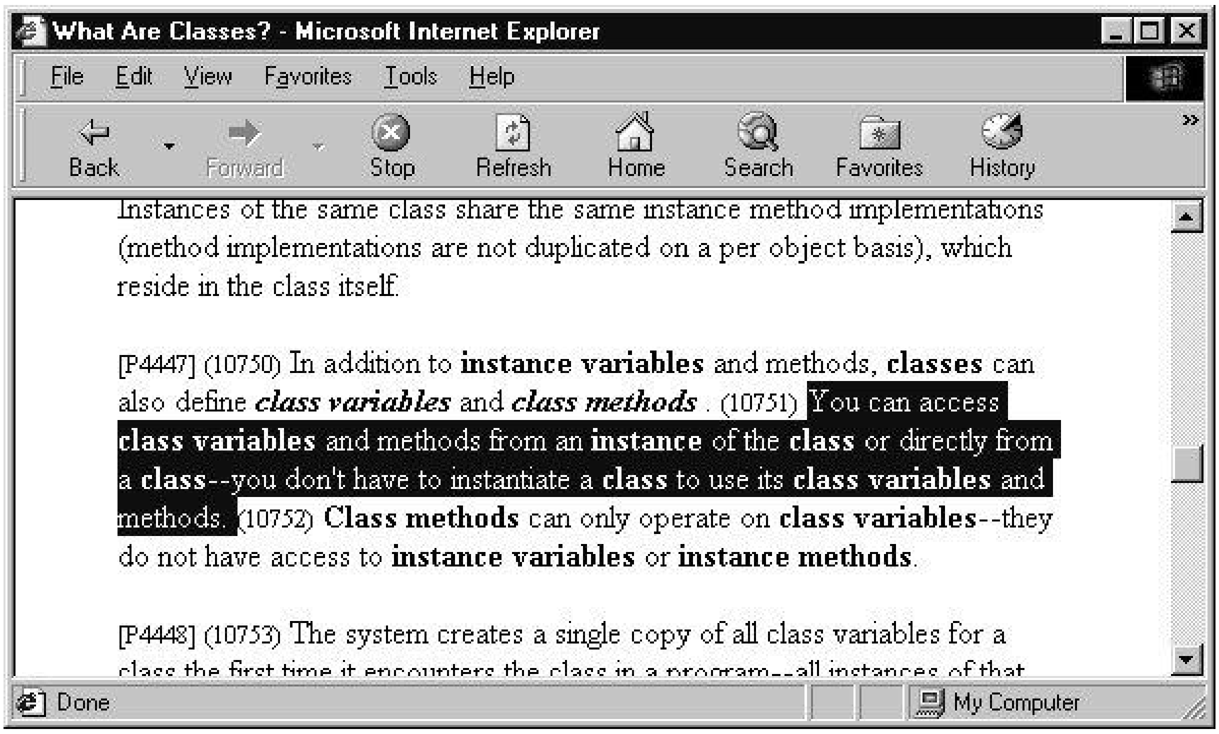
The *Knowledge Acquisition System* (KAS), which comes in two flavours:

1. *lite*: a simple interface usable by nonskilled people (we had a third year class use it successfully). It lets you pick out useful facts from a document collection searched by a conventional search engine (Glimpse, preceded by a sentence delimiter), and it displays only sentences, not pages. A few choices. eg which is the best fact, and you have a new tkb entry.

2. *professional*: a sophisticated concordancer+term extractor+parser etc designed to be used by skilled terminologists, with all the features they might dream of (Prof. Ingrid Meyer, a terminologist, has been particularly helpful in designing this system by having her students exercise and critique it.) For example, it understands syntax and knows semantic

patterns that express common relationships such as superordinate or part-of. It thus permits rapidly finding critical facts in a corpus. We used it to find about one-half the facts we put in the Java tkb. For example, you can ask it "what is a final method" or "what are the verbs that relate "class as subject to method as object?" and it will list and *rank* all sentences that it thinks relevant. Most are.

The *Knowledge organizer* (KO) is the actual knowledge management part of the system that stores, modifies, and displays the data in the tkb in a simple, restricted, tabular format. Figure 1 shows a part of a Java document while Figure 2 shows a typical KO screen. In Figure 1, we see some of the terms that are in the tkb in bold, and we see a fact highlighted in both the document and the tkb. Currently, the Java tkb has about 500 terms and about 1600 facts, the result of hundreds of person-hours of work. (We were beginners in Java.) About half of these facts were extracted from the associated document and the rest from other documents in our corpus using the KAS or from experts. Each fact entry points to its source.

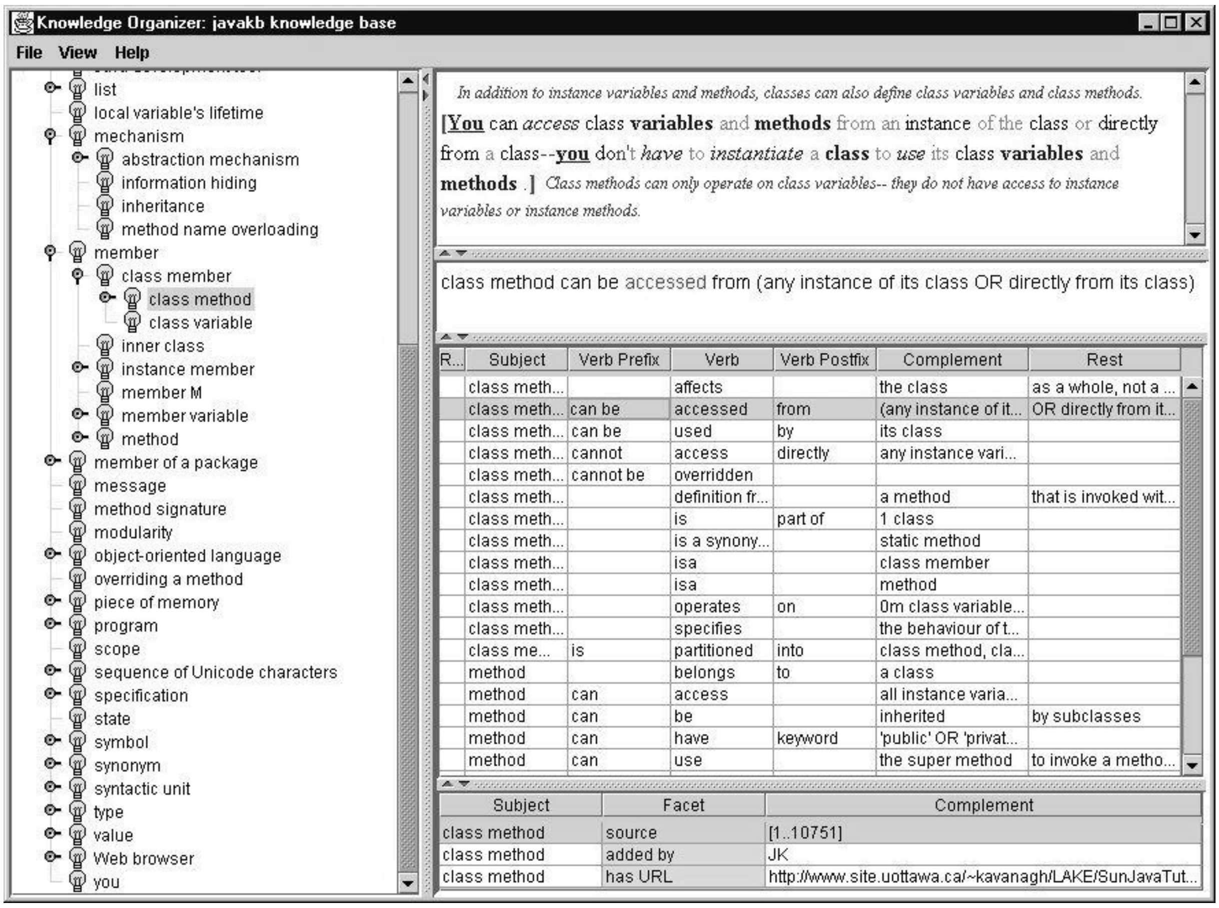

What Are Classes? - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History
Done
My Computer

Instances of the same class share the same instance method implementations (method implementations are not duplicated on a per object basis), which reside in the class itself.

[P4447] (10750) In addition to **instance variables** and methods, **classes** can also define ***class variables*** and ***class methods*** . (10751) You can access **class variables** and methods from an **instance** of the **class** or directly from a **class**--you don't have to instantiate a **class** to use its **class variables** and methods. (10752) **Class methods** can only operate on **class variables**--they do not have access to **instance variables** or **instance methods**.

[P4448] (10753) The system creates a single copy of all class variables for a class the first time it encounters the class in a program--all instances of that

Figure 1: A part of a page from the Sun Java Tutorial. The bold terms are in the tkb with associated facts, including definitions, and the highlighted text is a fact in the tkb under the subject **class variable** and **instance of a class**.



- list
- local variable's lifetime
- mechanism
  - abstraction mechanism
  - information hiding
  - inheritance
  - method name overloading
- member
  - class member
    - class method
    - class variable
  - inner class
  - instance member
  - member M
  - member variable
  - method
- member of a package
- message
- method signature
- modularity
- object-oriented language
- overriding a method
- piece of memory
- program
- scope
- sequence of Unicode characters
- specification
- state
- symbol
- synonym
- syntactic unit
- type
- value
- Web browser
- you

In addition to instance variables and methods, classes can also define class variables and class methods. **[<u>You</u>** can *access* class **variables** and **methods** from an instance of the class or directly from a class--**<u>you</u>** don't *have* to *instantiate* a **class** to *use* its class **variables** and **methods** .**]** *Class methods can only operate on class variables-- they do not have access to instance variables or instance methods.*

class method can be accessed from (any instance of its class OR directly from its class)

R...	Subject	Verb Prefix	Verb	Verb Postfix	Complement	Rest
	class meth...		affects		the class	as a whole, not a ...
	class meth...	can be	accessed	from	(any instance of it...	OR directly from it...
	class meth...	can be	used	by	its class	
	class meth...	cannot	access	directly	any instance vari...	
	class meth...	cannot be	overridden			
	class meth...		definition fr...		a method	that is invoked wit...
	class meth...		is	part of	1 class	
	class meth...		is a synony...		static method	
	class meth...		isa		class member	
	class meth...		isa		method	
	class meth...		operates	on	0m class variable...	
	class meth...		specifies		the behaviour of t...	
	class me...	is	partitioned	into	class method, cla...	
	method		belongs	to	a class	
	method	can	access		all instance varia...	
	method	can	be		inherited	by subclasses
	method	can	have	keyword	'public' OR 'privat...	
	method	can	use		the super method	to invoke a metho...

Subject	Facet	Complement
class method	source	[1..10751]
class method	added by	JK
class method	has URL	http://www.site.uottawa.ca/~kavanagh/LAKE/SunJavaTut...

Figure 2: The main screen of the Knowledge Organizer. The left pane shows the concept hierarchy. The term <u>class method</u> is selected. The bottom right pane shows facts as a database table. The top right pane shows the selected fact in full (XML-controlled) detail. The facets at the bottom hold any annotation one wishes to associate with the fact, eg its source.

3 Related Work

Euzenat (96) has a web-based system called HyTropes, which features a frame-like hierarchy linked to web documents. Martin and Ekland (99) describe an AI knowledge representation that provides fact access via the web. A number of papers presented at the Knowledge Acquisition Workshops (see those at KAW 1996 - 1999) present systems that also put knowledge bases on the web. A number of companies (e.g. Boeing, see Wojcik98, or Xerox) have been using controlled language for years, akin to how we enter facts into the tkb. Their main goal was, at least in the past, near-perfect machine translation. X facts certainly afford this opportunity. Other possibilies, for example, would include automatic translation into a formal logic system (such as Otter) for automatic reasoning.

4 Summary

It is important to understand the advantages of a dkb over a conventional knowledge resource such as a text book or a web-based document. A dkb offers:

- subjects that are arranged in *concept hierarchies,* which greatly aids understanding (they can also be arranged in other kinds of hierarchies, eg, part-of, as in Wordnet.);
- *ease of access*, eg browsing by subject, verb, other syntactic components, or graphically;
- the facts that are very *clearly expressed* and easy to find.

En bref, as the French say, X takes a step toward enhancing today's web documents by providing the reader with access to associated lexical and encyclopedic knowledge with one click. We do this by embedding controlled language facts in an XML document, and concurrently storing these as well in a special highly structured knowledge base for rapid retrieval and organization. If a document were already written in some kind of controlled language, as more and more will be, then it would be not too hard to almost automatically "translate" a sentence into X's format. (We estimate that a person with some language skills could assist this process at a rate of several facts per minute.) Facts in such a controlled language would, of course, be easy to translate automatically into other languages. Even more promising, they are amenable to all kinds of automatic machine language processing in a way that uncontrolled natural language still is not.

More information can be found at www.site.uottawa.ca/lake

References

Dieng, R., Corby, O., Giboin, A., and Ribière, M. (1998) *Methods and Tools for Corporate Knowledge Management* in KAW98.

Fensel, D., Decker, S., Erdmann, M., and Studer, R. *Ontobroker: Or How to Enable Intelligent Access to the WWW,* in: KAW98.

Euzenat J. (1996) *Corporate memory through cooperative creation of knowledge bases and hyper-documents.* in KAW96.

KAW 1996 - 1999 (1996 - 1999) Proceedings of the recent *Knowledge Acquisition Workshops*, Banff. (Gaines, B and Musen, M., eds).

Pratt, W., Hearst, M., and Fagan, L. (1999) A Knowledge-based Approach to Organizing Retrieved Documents. Proceedings of AAAI99.

Martin, P. and Eklund, P. (1999) Embedding knowledge in web documents. Proceedings of WWW Conference 99, Toronto.

Meyer, I. and Skuce, D. (1992). "Computer-Assisted Concept Analysis for Terminology: A Framework for Technological and Methodological research". *Proceedings of the Euralex Fourth International Conference* . Barcelona: Bibliograf, pp. 129-138.

Meyer, I., Eck, K., and Skuce, D. (1997). Systematic Representation of Concepts in a Knowledge-based System. In *Handbook of Terminology Management,* Eds. S. E. Wright and G. Budin. Amsterdam/Philadelphia: John Benjamins. pp 98-118.

Skuce, D. (2000) Integrating web-based Documents, Shared Knowledge bases, and Local Document Searching for User Help. *Computational Intelligence*, v16, no1. (February)

Skuce, D. (1995) Knowledge Management in Software Design: a Tool and a Trial. *Software Engineering Journal,* (Sept) pp. 183-193.

Skuce, D. and T. Lethbridge. (1995) CODE4: A Unified System for Managing Conceptual Knowledge. *International Journal of Human-Computer Studies,* v. 42: pp. 413-451.

Skuce, D. (1993). A System for Managing Knowledge and Terminology for Technical Documentation. Third International Congress on Terminology and Knowledge Engineering, Cologne, (14 pp).

Wojcik, R., Holmback, H., and Hoard, J. (1998) Boeing Technical English: An Extension of AECMA SE beyond the Aircraft Maintenance Domain. Proceeding of CLAW98, Pittsburgh.