DISTRIBUTED CONCEPT MAPPING COLLABORATION USING JAVA

Roberto A. Flores-Méndez
Knowledge Science Institute
University of Calgary
Alberta, Canada T2N 1N4
robertof@cpsc.ucalgary.ca

ABSTRACT
Concept maps are graphical representations used as a medium for communicating knowledge among individuals. Such suitability for knowledge representation has lead to the implementation of several concept mapping tools on personal computers. As computer networks evolved and became widely available as mechanisms for information exchange, concept mapping implementations also changed for supporting distributed environments. This demonstration will show an implementation of a concept mapping tool in Java. The current implementation will be presented as a viable solution for developing portable concept mapping tools on heterogeneous computer network environments, such as the Internet and the WWW.

1. INTRODUCTION
The notion of conveying ideas using graphical representations has been used throughout history as a means for expression and understanding among human beings. Of these graphical representations, concept map is one of the most widely used, due to its simplicity and explicitness.

With the advent of computers and networks, efforts were made to implement concept mapping tools as instruments for supporting distributed collaboration. At the Knowledge Science Institute (KSI), implementations under UNIX, Windows and Macintosh systems have provided an starting point for exchanging information among users of these platforms. However, this diversity of systems also required maintenance of code for each one of those platforms (which is not a minor task) until a workable option for portability was available.

Recently, the Java programming language (Sun, 1996) has been deployed as a feasible solution for this problem by providing multiple-platform execution of code, based on the use of intermediate 'bytecodes'. Regardless of its novelty, characteristics found on this programming language have made it a good candidate for producing portable implementations of distributed concept mapping tools.

2. CONCEPT MAPS
The term concept map, which encompasses a wide variety of diagrammatic knowledge representations, can be defined as diagrams composed of links and nodes of different types. Concept maps can be used to graphically represent and organize arguments and thoughts, thus providing an alternative to natural languages as a means to communicate knowledge (Gaines and Shaw, 1995a).

Figure 1 shows a concept map, constructed to visualize features found in the Java programming language. In this graphic circle nodes were used to represent concepts, rectangle nodes to represent instances, and labeled arrows to represent relationships.

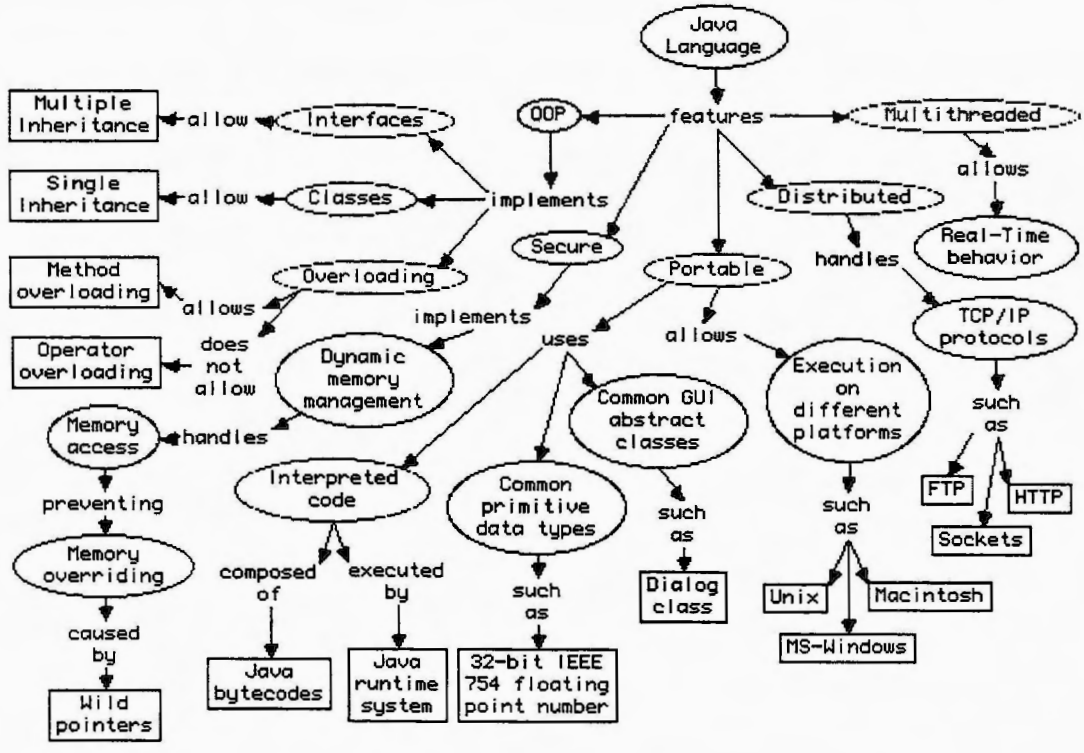

Figure 1. Concept map on the Java language.

3. COMPUTER-BASED CONCEPT MAPPING SYSTEMS

Due to their low cost and high performance, personal computers made possible the development of applications that allow interactive graphical constructs of concept maps. While a concept map system is a valuable tool for the single user, its potent rests on multi-user environments, where concept maps can provide user interfaces to shared resources, supporting a range of group processes such as brainstorming, collaborative planning, and join development of knowledge structures (Kremer and Gaines, 1994). Therefore, the use of a distributed platform was needed in order to support timely and geographically dispersed communities. As a natural choice, the Internet and the World-Wide Web (WWW) were chosen due to their communication infrastructure and multimedia capabilities.

At the KSI, efforts undertaken for developing such systems have resulted on the implementation of KSSn (Gaines, 1991), Accord (Kremer 1993), and XConMap (Lapsley, 1995), which are concept mapping tools designed for the Macintosh, MS-Windows and UNIX operating systems, respectively. As reported in (Gaines and Shaw, 1995c), (Kremer, 1996) and (Lapsley, 1995), these implementations were successfully integrated to the WWW as helper applications, Netscape plug-ins or Common Gateway Interface (CGI) server processes.

These developments cover much of the spectrum of installed computer platforms on the Internet, giving support for concept mapping tool users on those computer architectures. However, each implementation is tied to an specific operating system, a circumstance that requires work on three separate versions of the system, so their functionality can be maintain on each operating system. Definitely, a new implementation approach, a portable approach, was needed to attenuate the workload required for maintaining code developed on dissimilar computer platforms.

4. A CONCEPT MAPPING TOOL IMPLEMENTATION IN JAVA

As mentioned, a programming tool was needed for creating a portable version of the concept map tool and, additionally, with the ability of integrating with the WWW. While Java is not the only language to do so (Tcl/Tk (Ousterhout, 1994) may be used with similar results), this language was proclaimed as a panacea for deploying portable executable code on distributed environments. This assertion was mainly based on the features of a proprietary set of intermediate instructions called "bytecodes", which is Java's cornerstone for satisfying the promise of code "compiled once, run everywhere".

It was mainly due to marketing efforts that Java was widely adopted by the Internet community, specifically when popular WWW browser companies started to ship Java run-time interpreters as an integrated part of their products. Nevertheless, the outcome was an improvement of the multimedia capabilities of the WWW by allowing the integration of executable code with WWW documents, a fact which positioned Java as a promising tool for porting concept mapping tools to the WWW.

4.1. Class Hierarchy

Previous research accomplished at the KSI have resulted in the conforming of a class hierarchy suitable for implementing an open architecture concept mapping system. This hierarchy was constructed as a C++ class library and named KSImapper (KSI, 1996). The idea behind this hierarchy was to isolate the functionality of a concept mapping tool from representations inherent to specific operating environments, thus improving source code portability. Additionally, the software architecture was molded after design patterns specified in (Gamma, Helm, Johnson and Vlissides 1995), which facilitated (among other issues) the construction of a distributed system that can be kept in synchrony by utilizing commands generated by users and broadcast to each participant in a session (Kremer, 1996).

Figure 2 shows the KSImapper's core class hierarchy, already modified for the Java implementation, where ellipses represent abstract classes, rounded rectangles interfaces ,and squares instance-able classes.

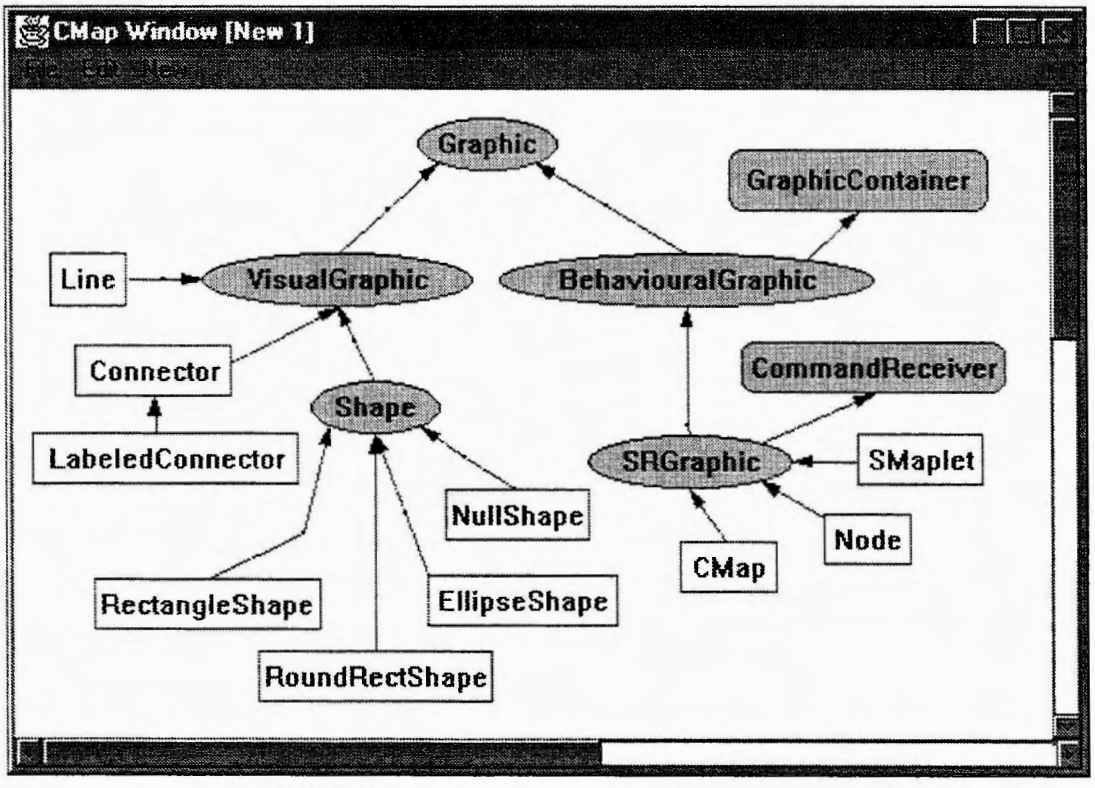

Figure 2. Core class hierarchy for implementing a concept mapping tool using Java.

Since Java's characteristics are similar, but not equal, to C++, the KSImapper class hierarchy required some slight modifications before been ported to Java. One of the modifications needed was due to the lack of multiple inheritance for non-abstract classes in Java. This circumstance forced some of the classes to implement methods that otherwise would be implemented in a parent class. For instance, in the original KSImapper C++ implementation the class `BehaviouralGraphic` has both the `Graphic` and `GraphicContainer` classes as parents, each with their own implemented methods. For the Java class hierarchy, one of these methods was required to be declared as an interface if both were intended to be parents of `BehaviouralGraphic`. For this case, `GraphicContainer` was chosen to be an interface, resulting on all of its methods been implemented on the class `BehaviouralGraphic`. Certainly, this kind of single-class-inheritance-and-interfaces structure is functional, but it may not be as practical as if multiple class inheritance was used.

Another change required was the substitution of pointers to dynamic memory, a characteristic that Java does not implement. For the purist C programmer, the lack of pointers may be seen as heretical, since very efficient code can be produced using them. However, it is also true that most of the bugs injected on programs are because of their use (MacGuire, 1993). Experiences obtained from this implementation have empirically shown that the development time tend to be more productive if pointer errors are not an issue.

4.2. Run-time System Architecture

The Java concept mapping system, which is conformed of a group of Java bytecode classes, may be executed as a Java application or as a Java applet. The differences between them are basically on the context of their use:

- Java applets are programs, integrated as part of WWW documents, which are downloaded and executed on the client computer when the document to which they are linked is invoked by a client WWW browser. Java applets require a WWW browser with a Java run-time interpreter built-in for them to be executed.
- Java applications are also programs, but they do not require the presence of a WWW browser for execution or downloading In that sense, Java applications are like any other regular stand-alone program that users may choose to install on their computer. Java applications require a Java run-time interpreter installed on the client computer.

Whether a concept mapping tool applet or application is executed, the scheme of execution is similar, as shown on Table 1 and Figure 3:

Applet	*Application*
*The WWW document containing a link to the concept mapping applet is requested by the client WWW browser.	*All Java classes for the concept mapping tool reside on the client computer.
*The Java classes for the concept mapping tool are automatically downloaded and executed by the WWW browser.	*Execution is started by invoking the Java run-time interpreter located locally at the client computer.
*A communication channel is requested to a process running on a predefined server hooked to the Internet. The communication channel will be used as the media for transmitting events, and the server process will act as a registration point, as a provider of up-to-date concept maps available at the beginning of a multi-user session, and as a broadcaster of events generated by users to the rest of the active clients in the session.	

Table 1. Scheme of execution of the Java concept mapping tool.

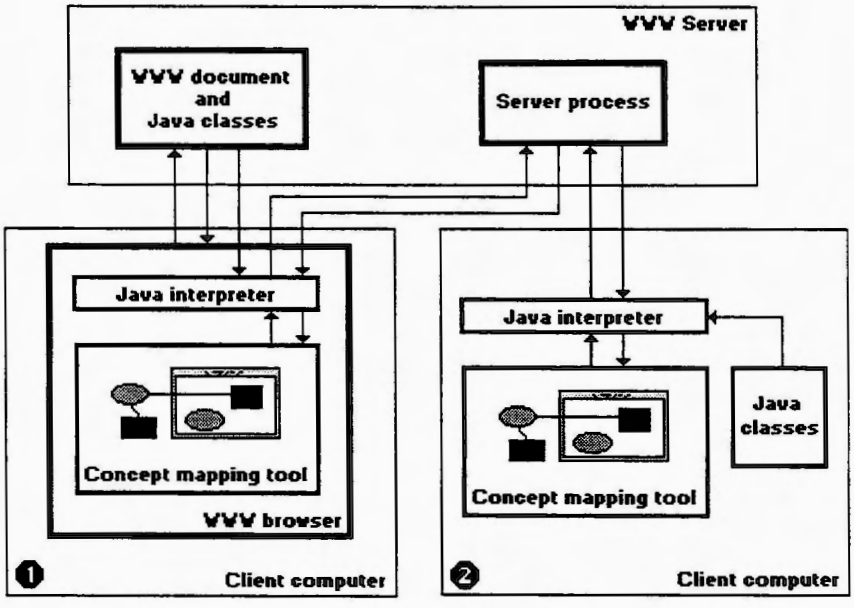

Figure 3. Run-time system architecture for Java applets ❶ and Java applications ❷.

5. FUTURE WORK

There is an unlimited number of options for the growth of the current implementation; for example, the run-time architecture may be augmented to handle several multi-user sessions (or channels), concurrency control or multi-user undo and redo (just supported on single-user versions); or the implementation may be extended to support formal representations or hypermedia for resource sharing, and so on. Eventually, these enhancements (and any others to emerge) will be evaluated as feasible options for extending the present implementation.

6. CONCLUSION

Concept maps are graphical representations used in multiple disciplines as a mean for communicating and sharing knowledge. With the advent of computers and networks, computer-based concept mapping tools were implemented to facilitate the manipulation of concept maps, not just for single users but for communities of individuals as well. During this demonstration, an implementation of a concept map tool was introduced as an example of the capabilities of Java for developing distributed collaboration environments on the Internet and the WWW.

7. REFERENCES

Gaines, B. R. (1991). *Organizational Modeling and Problem Solving using an Object Oriented Knowledge Representation server and visual language*, Proceedings of IJC-91, Sydney, Australia.

Gaines, B. R., and Shaw, M. L. G. (1995). *Concept Maps as Hypermedia Components*, Knowledge Science Institute, University of Calgary.
http://ksi.cpsc.ucalgary.ca/articles/ConceptMaps/

Gaines, B. R., and Shaw, M. L. G. (1995). *Collaboration through Concept Maps*, Knowledge Science Institute, University of Calgary.
http://ksi.cpsc.ucalgary.ca/articles/CSCL95CM

Gaines, B. R., and Shaw, M. L. G. (1995). *WebMap: Concept Mapping on the Web*, World Wide Web Journal, Fourth International World Wide Web Proceedings, December, 1995, O'Reilly & Associates, Inc.
http://ksi.cpsc.ucalgary.ca/articles/WWW/WWW4WM/

Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1995). *Design Patters: Elements of Reusable Object-Oriented Software*, Addison-Wesley, Reading Mass.

Kremer, R. (1993). *A Concept Map Based Approach to the Shared Workspace*, Masters Thesis, Department of Computer Science, University of Calgary.

Kremer, R. and Gaines, B. R. (1994). *Groupware Concept Mapping Techniques*, Knowledge Science Institute, University of Calgary.

Kremer, R. (1996). *Toward a Multi-User, Programmable Web Concept Mapping "Shell" to Handle Multiple Formalisms*, 10th Knowledge Acquisition Workshop Proceedings, Banff, Canada.

Knowledge Science Institute (1996). *CMap: KSI's Concept Mapping Interface Library*, Knowledge Science Institute, University of Calgary.
http://www.cpsc.ucalgary.ca/~kremer/CMap/index.html

Lapsley, A. Z. (1995). *Development of a Mediator System on the World-Wide Web to Model the Concurrent Manufacturing Life Cycle*, Master Thesis, Department of Computer Science and Department of Mechanical Engineering.

MacGuire, S. (1993). *Writing Solid Code*, Microsoft Press, 1993.

Ousterhout, J. (1994). *Tcl and Tk Toolkit*, Addison-Wesley.

Sun Microsystems (1996). *The Java Programming Language*, Sun Microsystems, Inc.
http://java.sun.com/