The K-Rep System Architecture

Robert Weida, Eric Mays, Robert Dionne, Meir Laker, Brian White, Chihong Liang, Frank J. Oles

IBM T. J. Watson Research Center*
P.O. Box 218
Yorktown Heights, NY 10598

1 Introduction

K-Rep is an object-oriented knowledge representation system based on description logic [Woods and Schmolze, 1992]. K-Rep provides a language to express descriptions (e.g., of medical concepts such as drugs, treatments, and diseases), inference mechanisms for reasoning about descriptions and their relationship to one another, and considerable supporting infrastructure. This paper presents an overview of K-Rep's system architecture, which has been motivated in part by the needs of several applications in the area of clinical information systems [Mays *et al.*, 1996].

It has become clear that accurate and comprehensive controlled medical terminologies are crucial for advanced clinical information systems, e.g., to achieve comparable data across the enterprise in support of outcomes analysis and inter-operation of ancillary systems, conceptual indexing of clinical advice, and knowledge-based entry of orders for drugs, procedures, etc. Organization and maintenance of such a terminology is a formidable task: there are at least 100,000 concepts of initial interest, both national (or international) standardization and local customization of content must coexist, and the terminology will evolve significantly over time. The state of the art in medical terminologies is characterized by *ad hoc* definitions, manual classification, little inferential power, and no provision for collaborative development. Therefore, the area of clinical information systems has become one of our key applications. In October 1994, after several years of experience with a Common Lisp implementation of the K-Rep description logic system [Mays *et al.*, 1991], we embarked upon a substantial redesign effort to accomplish the following objectives:

Scalability: K-Rep must support extremely large knowledge bases, potentially beyond the limits of virtual storage.

Persistence: Knowledge bases must be immediately available to applications, and must continue to exist independent of process termination or system shutdown.

Distribution: Knowledge bases must be simultaneously accessible from multiple client systems in a networked environment.

Embeddability: K-Rep must offer terminology services as a seamlessly integrated component of larger application systems.

Performance: K-Rep must be able to respond at interactive speeds in critical production environments.

Authoring support: K-Rep must provide a professional development environment to facilitate the collaborative knowledge engineering and visualization activities of domain experts in an intuitive and appealing manner.

Consistency maintenance: Revision and reclassification of descriptions is essential, both for interactive knowledge base editing and for run-time inferencing.

Scope: K-Rep must accommodate a wide variety of information within descriptions, including both semantic roles and primitives, and non-semantic information such as bitmaps, synonyms, abbreviations, and codes.

These considerations led us naturally to our current design point, illustrated in Figure 1 and discussed below.

2 Discussion

K-Rep addresses scalability and persistence using ObjectStore, an object-oriented database management system with a persistent memory model. Persistent memory extends virtual memory similar to the way that virtual memory extends real memory, but also includes the database characteristics of atomic, serializable, and recoverable transactions. A particularly important feature of ObjectStore's persistent memory implementation is the mapping of persistent memory pages into virtual memory pages using *pointer swizzling* [Lamb *et al.*, 1991]. For example, once a persistent page's addresses have

*The first author may be contacted via email at *weida@cs.columbia.edu*. The other authors, respectively, may be contacted at {*emays, dionne, meir, bfwhite, cliang, oles*}*@watson.ibm.com*.

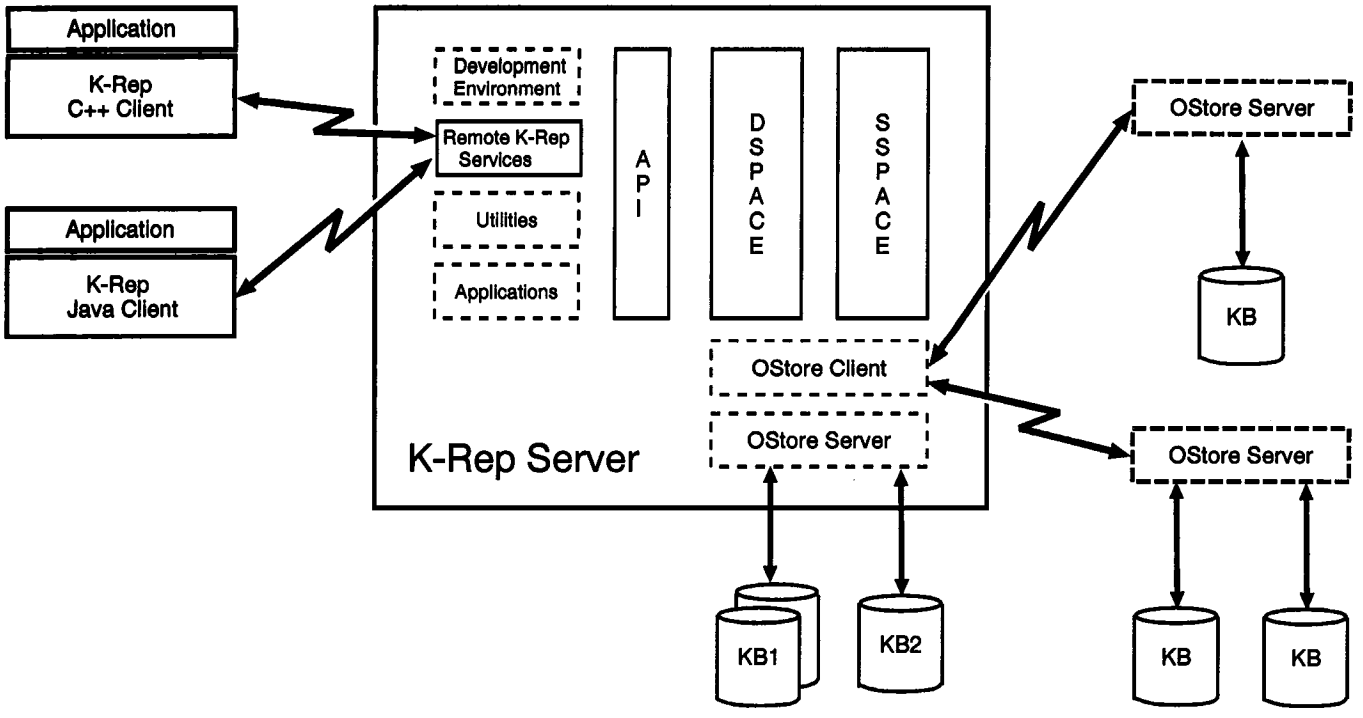

Figure 1: System Architecture

been mapped into virtual memory, dereference of a persistent pointer to that page is implemented with a single *load* instruction. An address reference may occur outside of the mapped range, in which case a persistent page fault occurs. K-Rep uses segment allocation and clustering to improve locality of reference, and its algorithms are specially designed to minimize persistent memory references. Although relational database management systems are now evolving to support objects in the form of BLOBS[1], contemporary systems are simply unable to match ObjectStore in terms of speed. Nonetheless, we are actively exploring appropriate uses of relational technology as our framework evolves.

C++, however grotesque, was an obvious pragmatic choice for our core implementation language due to its object-orientation, portability, and amenability to integration with database systems and applications. (Java's cleaner design, potentially superior portability, and especially its automatic memory management make it an intriguing future alternative.) K-Rep's C++ class library defines an application program interface (API) which presents an interface to the underlying kernel. The API encapsulates numerous implementation details such as the persistent storage mechanism. It also protects the integrity of internal data structures, including all those which persist. An identical interface is presented by an ephemeral (or transient) memory version of K-Rep which does not use ObjectStore. Using ephemeral knowledge bases consumes fewer system resources and facilitates both application development and development of smaller knowledge bases. In addition to K-Rep applications, both the K-Rep development envirommment and a set of K-Rep system utilities access the kernel strictly through the API.

The K-Rep kernel is divided into *definition space* (DSPACE) and *semantic space* (SSPACE) components, which are responsible for managing descriptions and their classification, respectively. The DSPACE maintains consistency among several representations of a description:

1. A *source form* is a character string which is suitable for text file import/export operations, and can be replaced through an API call, e.g., when using the development environment's text editor.
2. A *normalized form* is a canonical, recursive data structure for the description *per se*, which is suitable for tabular presentation and structural editing. It is updatable through API calls, e.g., to implement "direct manipulation" editing in the graphical development environment. Among other things, the normalized form includes both *normalized roles* for semantic attributes and *facets* for non-semantic attributes of concepts and roles as exemplified earlier.
3. A *completed form* captures the description's meaning in the overall context of the knowledge base, i.e., it incorporates the results of inferences such as inheritance. It is only updated indirectly through changes to source or normalized forms.

[1]Binary Large Objects.

There is a many-to-one mapping from DSPACE concepts to their counterparts in the SSPACE. Taxonomic relationships among DSPACE concepts are determined by reference to the corresponding SSPACE concepts, which are organized in an explicit subsumption taxonomy. This approach neatly handles the coexistence of equivalent concepts. Semantic descriptions only record local differences from immediate ancestors in the taxonomy. We take advantage of this to reduce the number of tests required by our classification algorithm, and to to minimize the size of SSPACE concepts. In the context of ObjectStore, the DSPACE/SSPACE separation provides dramatic performance benefits: SSPACE concepts are clustered in separate ObjectStore segment(s), so only SSPACE pages are mapped during classification (and only for those SSPACE concepts encountered while traversing the taxonomy). In addition, the small size of SSPACE concepts reduces the number of pages accessed.

K-Rep can operate in either single system or distributed client/server mode. Distribution, in turn, is achieved in two different ways, either through ObjectStore's client/server facility using K-Rep's native C++ API, or with K-Rep's own client/server query facility, where a run-time server provides access to K-Rep for lightweight clients via a scripting language (parameterized queries can be parsed and registered for efficient reuse). The use of a scripting language minimizes the number of separate client/server transactions, and enables desktop applications where system resources are at a premium (as may be the case in a clinical workstation). Both C++ and Java clients are implemented using the scripting language. The server provides a range of lexical pattern matching capabilities which desktop clients may use to locate terms. A desktop client can also navigate the structure of a knowledge base to locate items by their classification. Various forms of navigation are possible depending on the user interaction technique.

K-Rep's development environment includes a direct manipulation graphical user interface, which features browse, query, and edit capabilities, a commit/abort protocol for modifications on a per concept basis (orthogonal to database transaction commits/aborts), and a journal facility. Considerable thought and effort was required to make the user interface suitable for serious knowledge engineering activities. A *taxonomy viewer* allows users to navigate the taxonomy in an incremental, top-down manner. A user can focus his or her attention by selectively revealing and hiding portions of the taxonomy. Other viewers, such as the *definition viewer* which presents the normalized form and the *significance viewer* which presents the completed form, help users to assimilate definitions, their semantic import, and the relationship between the two. Incremental terminology development is fostered by a variety of mechanisms for description "copy and edit" and role restriction refinement.

During terminology development, K-Rep operates on a database in exclusive write mode, because modifications may have non-local effects. In the same way that design systems support long running transactions, K-Rep has a model of collaborative work. It records a journal of changes as textual representations. The journals associated with several terminology developers may then be examined and merged. A methodology for automating this process in association with K-Rep is the subject of [Campbell *et al.*, 1996]. Key concerns in concurrent, distributed development of terminologies include:

- Conflict detection and resolution
- Distribution of approved terms
- Update of local versions

Of course, subsumption and related inferences are extremely useful for conflict detection and reporting.

3 Conclusion

Bringing K-Rep to the point of around-the-clock production use at Kaiser-Permanente in Colorado [Mays *et al.*, 1996] has been a non-trivial endeavor. Other projects are underway at the inter-regional level of Kaiser-Permanente, Barnes Jewish and Christian Hospital, Columbia Presbyterian Medical Center, the Mayo Clinic, and Stanford University. In our experience, rigorous attention to a wide range of "systems issues" is crucial, both to elicit interest from development partners and commercial customers, and to win their acceptance.

References

[Campbell *et al.*, 1996] K. E. Campbell, S. P. Cohn, C. C. Chute, G. Rennels, and E. H. Shortliffe. Galapagos: Computer-based support for evolution of a convergent medical terminology. In *Proceedings of the 1996 AMIA Annual Fall Symposium*, Washington, DC, 1996. To appear.

[Lamb *et al.*, 1991] C. Lamb, G. Landis, J. Orenstein, and D. Weinreb. The objectstore database system. *Communications of the ACM*, 34(10):50–63, Oct 1991.

[Mays *et al.*, 1991] E. Mays, R. Dionne, and R. Weida. K-rep system overview. *SIGART Bulletin*, 2(3):93–97, June 1991. Special Issue on Implemented Knowledge Representation and Reasoning Systems.

[Mays *et al.*, 1996] E. Mays, R. Weida, R. Dionne, M. Laker, B. White, C. Liang, and F. J. Oles. Scalable and expressive medical terminologies. In *Proceedings of the 1996 AMIA Annual Fall Symposium*, Washington, DC, 1996. To appear.

[Woods and Schmolze, 1992] W. A. Woods and J. G. Schmolze. The kl-one family. *Computers and Mathematics with Applications*, 74(2-5), 1992.