An Object-Oriented Software Architecture for the Explorer-2 Knowledge Management Environment

David B. Tarabar, M.S.
Robert A. Greenes, M.D., Ph.D.
Eric T. Slosser

Harvard Medical School, Decision Systems Group
Brigham and Women's Hospital, Department of Radiology
Boston, Massachusetts

Abstract

Explorer-2 is a workstation based environment to facilitate knowledge management. It provides consistent access to a broad range of knowledge on the basis of purpose, not type. We have developed a software architecture based on Object-Oriented programming for Explorer-2. We have defined three classes of program objects: Knowledge ViewFrames, Knowledge Resources, and Knowledge Bases. This results in knowledge management at three levels: the screen level, the disk level and the meta-knowledge level. We have applied this design to several knowledge bases, and believe that there is a broad applicability of this design.

1. Introduction

Explorer-2 is a workstation based environment to facilitate *knowledge management*. By this we refer to the set of tasks involving (1) selective retrieval of knowledge for problem solving, (2) browsing and navigation through available knowledge bases, and (3) annotation and tailoring of pathways through knowledge bases for personal storage and subsequent access. [6]. Explorer-2 is a successor to Explorer-1 [3]. Explorer-1 is a knowledge management system that primarily supports hypertext and keyword-based access to knowledge, along with limited access to tabular data and simulations. Explorer-2 extends our earlier work by using a consistent approach to supporting access to a broader range of knowledge types. In Explorer-2 the linking and communication among knowledge units is generalized. Our view is that knowledge is accessed on the basis of purpose, not type [5]. We wish to provide seamless access to lists, outlines, hypertext, images, or dynamic knowledge in the form of simulations, analyses or inferencing.

This paper describes the software design that we have developed for the implementation of Explorer-2. Experience with Explorer-1 has influenced this design — both user and developer experience. We have based this design on a software technology, Object-Oriented Programming, that seems appropriate to the current and future task of implementing Explorer-2. We will define some basic concepts of knowledge management and then describe a software design that we have used to realize these concepts.

An aim of Explorer-2 is to provide a highly interactive graphical environment for medical "knowledge workers" on a widely accessible and affordable workstation. We have implemented Explorer-2 on the Macintosh® family of personal computers (Apple Computer, Inc.), utilizing its multi-window, event driven programming style. As our implementation language, we have used Object Pascal [1], a set of Object-Oriented programming extensions to Pascal developed by Apple.

2. The lessons learned from Explorer-1

In our evaluations of Explorer-1, we noted a generally favorable response to the concept of electronic access to medical knowledge. This enthusiasm was tempered by the limits of the knowledge available through Explorer-1 [4]. Authoring of hypertext content for Explorer-1 was primarily done using standard word processing programs, and then running a compiler which imported the content into Explorer-1 files and created links. Interactive author editing and establishment of links while browsing through content was not supported — which made the content generation process cumbersome.

One of the principal features of Explorer-1 is the support of several navigation aids. Multiple overlapping windows of content can be displayed concurrently. A "desk" menu provides a list of opened windows and allows any one to be selected and brought to the front. An "overview map" window displays the pathway of connectivity that has been followed by the user in navigating the knowledge base. For any knowledge unit (corresponding to a window), a list of potential links to/from that window can be displayed and a link of interest may be selected and followed. Nonetheless, with many windows opened concurrently, a more effective display of *context* was also considered desirable to maintain user orientation. This includes both a modeless overview map which would not take over the screen, and also direct context information on a per window basis.

The initial implementation of Explorer-1 was done on a 512K Macintosh — a machine that seems quaintly underpowered by today's standards. The early Macintosh provided a clear vision of the graphic user interface, but required great effort on the part of the developer to squeeze something useful into such a small machine. As a result, the programs were not designed for growth, but rather they were designed to fit. As Macs became more powerful, we were able to add functions to Explorer-1, but it became more and more difficult. As our wish list increased it became necessary to do a reimplementation. Explorer-2 duplicates and greatly

0195-4210/89/0000/0093$01.00

expands the functionality of its predecessor. It has been designed with a software architecture that supports flexibility and growth.

3. Object-Oriented programming

Object-Oriented Programming (OOP) [2,9] is a software technology with roots in the late 1960's. It has become increasingly popular in the 1980's — influencing the design of system software for workstations and spawning new variants of procedural programming languages. An Object-Oriented Program is organized as a set of intelligent data *objects* which contain both data and procedures (called *methods*) to manipulate that data. The program is responsible for sending *messages* to objects. These messages will invoke methods to accomplish the desired computation.

The power of OOP comes from two factors. The first is the modularity inherent in the class definition mechanism. In Object Pascal, one defines an object type (or *class)* with an extension of a Pascal record definition. Besides the normal variables defined in a record, procedures are also defined in an object. This object definition serves to insulate the internal structure of an object from the rest of the program. This results in an appropriately modular program design. Object classes may be defined in a hierarchy. This provides the greatest leverage of OOP — object inheritance. When you define an object class in terms of another class, the descendant *inherits* all of the data and methods of its parent. The *subclass* may add new variables or new methods, and it can *override* the implementation of existing methods. This allows one to implement a subclass by concentrating only on the new features of the object, while using already debugged methods when appropriate. A well designed scheme of class inheritance can make a program very extensible. It also facilitates group collaboration on large development projects, a feature that we have found extremely valuable.

4. What is a Knowledge ViewFrame ?

Inasmuch as we set out to build a knowledge management tool, we will describe here what we have found to be a useful pragmatic definition of knowledge. In general, knowledge is undefinable, but since we are working in the context of a computer workstation, we can venture the definition of a *Knowledge ViewFrame* (KVF) as an arbitrary unit of knowledge that appears in a window on a computer screen. It is important to realize that the human experience of this knowledge is not limited to its visual presentation, but also by interacting with it. The interaction may take the form of menu commands, keyboard actions or direct manipulation with a pointing device. The Knowledge ViewFrame may interact via procedural computation, animation or even sound. The net result of this visual and physical interaction is to transfer to a person some stored body of knowledge. Thus Explorer-2 is a knowledge management tool in that it supports the creation and manipulation of Knowledge ViewFrames.

The methods of a KVF support all of the standard operations of the desktop metaphor: resizing, moving, overlapping and updating. Optional methods support user interaction via keyboard or mouse input. Subclasses of the standard KVF may implement interactions such as linking, list expansion or whatever effectively transmits knowledge to the user.

5. What is a Knowledge Resource ?

The world's medical knowledge is not currently organized into the KVFs that we have defined. Machine readable knowledge bases typically exist as some mass of disk storage in one of many storage formats. We will define a *Knowledge Resource* (KR) as the program object that is aware of a single type of storage format and is able to create Knowledge ViewFrames on command. There are two properties of a specific disk format: (1) the storage mechanism that is used, e.g. ISAM, sequential files, custom database; and (2) the specific layout of records that are retrieved. A Knowledge Resource object implements these details and consists of methods that will create Knowledge ViewFrames. In Explorer-2, new KVFs are generated either by menu commands, global searches or via linking from a current KVF.

In the most simple case, we can consider a KR to be a database, where the creation of a KVF involves retrieving stored data and the formatting of it for display. However we see a Knowledge Resource as being much more flexible than that. Knowledge ViewFrames may be the result of extensive computation on stored data, so that the KVF is never actually stored on disk but is computed on request. A procedural Knowledge Resource may have no stored data, but only code that can perform some simulation or analysis in response to user input. In addition, a single Knowledge Resource may be used to produce several different 'types' of Knowledge ViewFrames. The wide range of possibilities that can be included under the Knowledge Resource abstraction is a key to the extensibility of Explorer-2.

6. What is a Knowledge Base ?

As we have stated earlier, Explorer-2 accesses knowledge on the basis of purpose, not type. Conceptually, we consider a Knowledge Base to be a collection of related information about a certain subject. We will define the *Knowledge Base* (KB) as the program object that captures the general properties of a related collection of knowledge. In Explorer-2, a KB always consists of one or more KRs. Notice that this means that the same KR may be a part of several KBs — thus there is no need to duplicate an existing KR if you are preparing a KB for a different purpose.

The methods of a Knowledge Base implement the global operations available on a KB. Explorer-2 can open and close a KB; display a description of a KB; execute top level menu commands of a KB; or search for KVFs based on keywords or relatedness between them. The implementation of these methods will in turn call methods of its KRs. Opening a KB involves opening the KRs that are a part of it. Menu commands will usually be dispatched to the appropriate KR.

7. Explorer-2 levels of Knowledge Management

The functionality of Explorer-2 can be divided into three levels: manipulation of KVFs; selection and manipulation of KBs/KRs; and what we call meta-knowledge utilities, as depicted in Figure 1.

On the first level, Explorer-2 is a program that manipulates Knowledge ViewFrames as resizeable, movable and overlapping screen windows on a interactive graphic workstation. The program supports the desktop metaphor and most importantly it directs user input to the topmost KVF in order to trigger the interactive possibilities of the KVF. A user can organize the available screen space to display the collection of Knowledge ViewFrames that best serve the purpose of the interaction.

On the second level, Explorer-2 provides an interface between disk storage and KVFs through the abstractions of Knowledge Resources and Knowledge Bases – thus allowing KBs and KRs to be prepared at multiple sites and and exchanged. The user of Explorer-2 has potential access to a universe of Knowledge Bases. The program provides features to select from the available KBs for personal needs. On this level, users can open and close KBs, use menus to retrieve Knowledge ViewFrames and search the available viewframes of a KB by keyword or some other relationship.

Most importantly, Explorer-2 provides search and navigational aids for the mass of knowledge that becomes available at the desktop. With all of the flexibility provided, there is a danger of getting lost in the knowledge universe. The overview map is perhaps the most powerful feature — it maintains a view of the pathway that the user has taken and allows the user to inspect (and select) the potentially available links related to a given KVF. In addition, the map may be edited and saved as a customized pathway through KVFs that pertain to a specific topic. These state maps may later be retrieved to recreate the desktop, with these specific KVFs. A Desk menu allows the user to select an obscured window by its title and bring it to the front. Finally, keyword search tools allow the user to find the appropriate KVFs based on keyword or pseudo-natural language query, instead of being constrained to the author-supplied structure.

8. Appropriateness of Object Oriented Design

Our design emphasizes Knowledge ViewFrames as the focal point of Explorer-2. We believe that the Knowledge ViewFrame is an appropriate abstraction for the purposes of Knowledge Management. By combining information and interaction, we may take advantage of the capabilities of the computer that are not present in existing books and journals. Text presented in a sequential and static manner will never improve upon the printed page. By adding interaction, we can build an active knowledge manager.

Our definition of a Knowledge ViewFrame echos the data and procedure of a program object and thus suggests an object-oriented programming approach. Using inheritance we can develop a hierarchy of KVFs that can share code as much as possible. We have also developed object definitions for KBs and KRs. OOP provides robustness due to the modularity that it enforces. It also provides extensibility due to the modularity and inheritance capabilities. Modularity is important, since Explorer-2 represents contributions from several members of our research group and we expect further work to be done by others.

We should also be more precise about what is meant by the 'type' of KVFs, KRs and KBs. In all cases we mean nothing more than that there is an object class which implements that 'type'. Deciding what should make up a type is solely a software design issue. The only constraint on defining a type is the desire or need to reuse it in the object hierarchy.

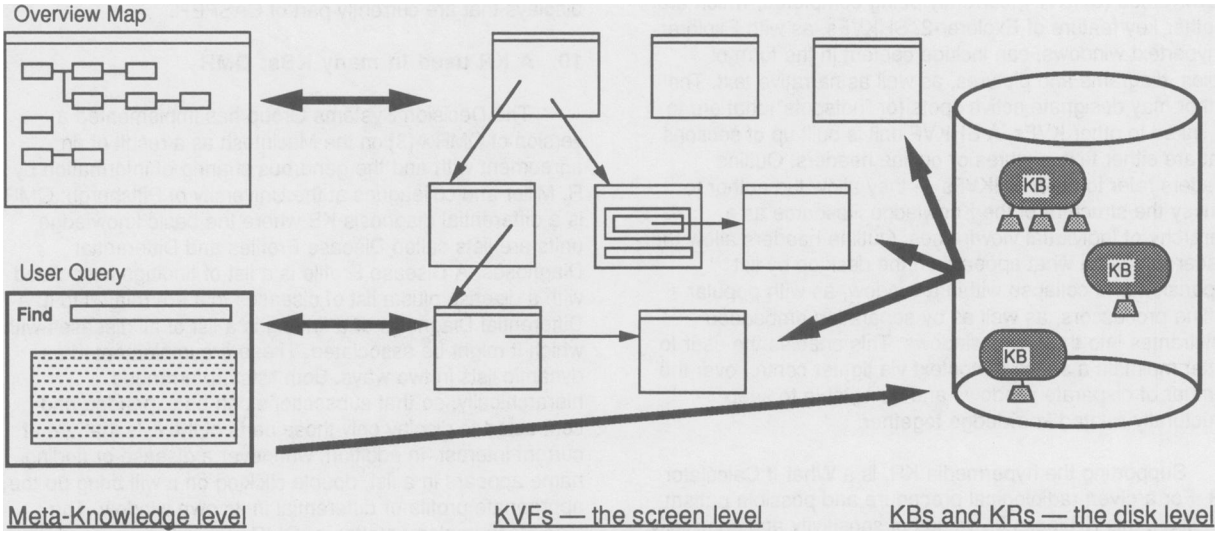

Figure 1: Relationships among the three levels of Knowledge Management in Explorer-2

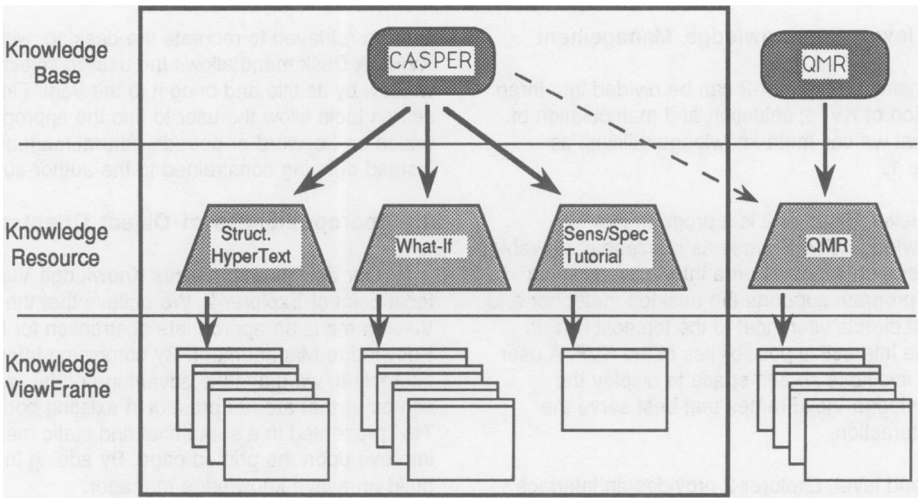

Figure 2: A block diagram of two Knowledge Bases and how they fit into our Software Architecture

9. A KB composed of many KRs: CASPER

The CASPER [4] KB was developed for Explorer-1 as an aid to diagnostic workup selection involving imaging procedures. It was originally implemented in hypertext form derived from a radiology diagnostic workup strategy handbook [7] — along with procedural displays that were added in an ad hoc manner. The design of Explorer-2 was motivated in large part by a desire to provide a formalized way to provide these multiple knowledge formats. The resulting architecture provides us an opportunity to recast this knowledge base using a wider variety of KVFs and KRs and to link with other existing KRs.

In our design, CASPER contains three Knowledge Resources (see figure 2). Once again, the bulk of the information is in hypertext form, as a "hypermedia KR", but we will use the "Structured HyperMedia" Knowledge ViewFrames (SHKVF), currently being completed, which are another key feature of Explorer-2. SHKVFs, as with Explorer-1 hypertext windows, can include content in the form of tables, diagrams and pictures, as well as narrative text. The author may designate active spots (or "hotspots") that are to be linked to other KVFs. A SHKVF unit is built up of sections that are either text, pictures or outline headers. Outline headers refer to other SHKVFs — they allow the author to convey the structure of the Knowledge Resource as a hierarchy of individual viewframes. Outline headers allow the reader to control what appears on the desktop by list expansion and collapse within a window, as with popular outline processors, as well as by separating embedded viewframes into their own windows. This enables the user to better maintain a sense of context via tighter control over the number of disparate windows and being able to keep structurally related knowledge together.

Supporting the hypermedia KR, is a What-If Calculator KR. For a given radiological procedure and possible patient condition, this KR looks up values of sensitivity and specificity. The What-If Knowledge ViewFrame then presents a spreadsheet-like decision support tool. The physician then may modify the pre-test probability, as well as rule-in and rule-out thresholds. There are links to specific What-If KVFs throughout the main body of Casper.

A third Knowledge Resource contains a Simulation that displays both numerically and graphically the meanings of Sensitivity, Specificity and Predictive Value. This KR contains a single KVF to which linkage can also be made.

We have discussed three KRs that we are currently building to create the new version of CASPER. The architecture we have presented also provides the potential of using other Knowledge Resources that are developed independently. Future KRs could include bibliographic reference files which could be searched to provide access to the references currently embedded in CASPER. Another KR that we may incorporate is a structured description of clinical workup strategy that can be used to dynamically generate a clinical algorithm KVF, rather than the static algorithm displays that are currently part of CASPER.

10. A KR used in many KBs: QMR

The Decision Systems Group has implemented a version of QMR® [8] on the Macintosh as a result of an agreement with and the generous sharing of information by R. Miller and colleagues at the University of Pittsburgh. QMR is a differential diagnosis KB where the basic knowledge units are lists called Disease Profiles and Differential Diagnoses. A Disease Profile is a list of findings associated with a disease plus a list of diseases that are related to it. A Differential Diagnosis of a finding is a list of all diseases with which it might be associated. These are implemented as dynamic lists in two ways. Both lists are arranged hierarchically, so that subsections can be expanded and contracted to display only those parts of the lists that are of current interest. In addition, whenever a disease or finding name appears in a list, double clicking on it will bring up the appropriate profile or differential in its own window. The Macintosh implementation of QMR was initially a standalone application, but we were able to create an Explorer-2 KB that would provide the same functionality.

In our design, the QMR KVFs are generally implemented as windows that contain QMR type lists where double clicking will do either list expansion or contraction. The QMR KR is responsible for the details of storage. The QMR database is stored in a custom storage format that consists of tens of files. The QMR KR contains code that extracts data from these files to build Disease Profiles or Differential Diagnoses or perform other functions, on demand. At present the QMR KB contains the single QMR KR and makes it selectable via the Open KB command of Explorer-2.

Our architecture also allows the QMR KR to be a part of other Knowledge Bases (see figure 2). Thus another KB, such as CASPER, might include the QMR KR in order to reference specific Disease Profiles. Once these profiles appear as Knowledge ViewFrames on the desk, the user can access the full functionality of QMR KVFs.

11. Future Directions: Explorer-n ?

We have presented the Object-Oriented software architecture that is being used in the implementation of Explorer-2. While this architecture has been chosen for the immediate task at hand, we have kept the future in mind. We believe that the abstractions of Knowledge ViewFrames, Knowledge Resources and Knowledge Bases have a more general applicability. Explorer-2 has been inspired by a history of visions of Computer Science such as Vannevar Bush's Memex and Alan Kay's Dynabook and has many of the features envisioned by the "Knowledge Navigator" propounded by Apple Computer.

In the near term, as we add more KR and KVF types to the repertoire of Explorer-2, we will be faced with a single application that will grow huge. At present this is necessary due to the lack of robust multiprocessing and interprocess communication in the Macintosh operating system. This situation is clearly undesirable and we would hope to see Explorer-2 evolve into several cooperating modular applications. Most of these applications would manage individual KB types, while others will provide the navigational aids that are found in our meta-knowledge utilities.

Eventually we would hope that the desktop metaphor would be extended to provide knowledge management as an expected service. Alongside the disks, folders and documents on ones desk, would be Knowledge Bases and Knowledge ViewFrames. It should be no more difficult to reference personal knowledge bases than to scan filenames. You would expect to find elaborate desk-level scripting mechanisms that would link together KBs and applications.

Acknowledgement

This work was partially supported by Grants LM 03707 and LM 04572 and Contract LM 635234 from the National Library of Medicine, by Grant CA 45574 from the National Cancer Institute, and by the CAMDAT Foundation, Inc.

We gratefully acknowledge our co-workers: Larry Cope, Stephan Deibel, William Hersh, MD and Jan Snydr-Michal for their assistance in the development and implementation of Explorer-2.

References

[1] Apple Computer, Inc.; Macintosh Programmer's Workshop Pascal, Version 3.0; (APDA #M0021LL/A); Cupertino, CA; 1988

[2] Cox B.; Object-Oriented Programming: An Evolutionary Approach; Addison-Wesley; 1986

[3] Greenes R.A.; Knowledge Management as an Aid to Medical Decision Making: The Explorer-1 System; Proc MEDINFO 86, North-Holland: Elsevier Publishers B.V., 1986: 895-899

[4] Greenes R.A., Tarabar D.B., Krauss M., Anderson G., Wolnick W.J., Cope L., Slosser E., Hersh W.; Knowledge Management as a Decision Support Method: A Diagnostic Workup Strategy Application; *Comput Biomed Res*, 22, 113-135 (Apr 1989)

[5] Greenes R.A.; "Desktop Knowledge": A New Focus for Medical Education and Decision Support; Proc Medical Informatics and Education International Symposium, May 1989, 89-96; also in *Meth Inf Med*, 1989 (in press)

[6] Greenes R.A., Tarabar D.B., Cope L., Slosser E., Hersh W., Pattison-Gordon E., Abendroth T., Rathe R., Snydr-Michal J.; Explorer-2: An Object-Oriented Framework for Knowledge Management; Proc MEDINFO '89 (in press)

[7] McNeil B.J. and Abrams H.L., eds.; Brigham and Woman's Handbook of Diagnostic Imaging; Little, Brown, and Co., 1986

[8] Miller R.A., McNeil M.A., Challinor S.M., Masarie F.E., Myers J.D; The INTERNIST-1/QUICK MEDICAL REFERENCE Project — Status Report; *West J Med,* 145, 816-822 (Dec 1986)

[9] Thomas D.; What's in an Object?; *Byte*, 14:3, 231-240 (Mar 1989)