A Human-Computer Interaction Framework for Media-Independent Knowledge (Position Paper)

Robert J.K. Jacob
James G. Schmolze

Department of Electrical Engineering and Computer Science
Tufts University
Medford, Mass. 02155

Abstract[1]

This position paper looks toward the future possibility of storing knowledge in a way that is independent of any media and modalities and of producing a variety of presentations and interactive displays in different media and multi-media combinations, all from the same stored knowledge. Different users may have personal preferences for some modes of interaction, different learning styles that favor one mode over another, or disabilities that prevent the use of some modes. The general problem of storing modality-independent knowledge and, from it, generating rich and compelling interactive displays in a variety of forms is unsolved. In this position paper, we begin to attack it by dividing the problem into pieces, proposing a framework within which progress might be made, and describing some partial solutions as a starting point for research.

Introduction

A computer might display information about how to repair a machine or a summary of daily stock market performance in a variety of media. It could present pictures or diagrams, animated video clips, a text document, a spoken lecture or narrative on the subject, or various multi-media combinations of these, such as a diagram with a spoken narration about it.

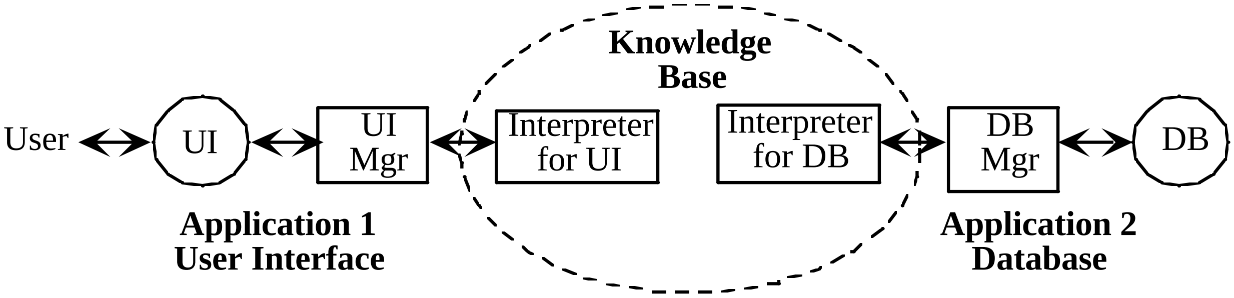
To realize this today, each of the separate representations of, for example, how to repair the machine must have been stored in the computer individually ahead of time. The video clips, the spoken lecture, and the combination description must each be input and stored separately in the computer, to be replayed on command. Some ad-hoc translation from one medium to another may be possible, such as extracting still pictures from a video or creating speech from a text file by speech synthesis. But such translations often result in presentations that are suboptimal in their new media. Information is lost in the translation, and other information that might be more appropriate in the target medium was not present in the source.

Instead, we envision a day when much of the knowledge about how to repair the machine might be stored once, in a form that does not depend on the choice of media used, and then output in different media and forms as needed to suit the individual user and the situation. The user may have personal preferences as to which media or combinations are preferred, or learning styles that work better or worse for different individuals, or disabilities—a blind user might use a spoken or other auditory display instead of a visual one. The playback situation may also require different modes—the user could be sitting in front of and watching a screen or driving a car, in which case an auditory presentation would be preferred.

Moreover, there will always be media-specific information, such as movies or sound recordings, that will play a role in multi-media interactions. These must be included in such a way that they can easily be integrated with interactive displays that are generated primarily from media-independent knowledge. Following Maybury and Wahlster (Maybury, 1998), we can speak of media (video, audio, text), which are used to capture, store, and/or transmit encoded information, modalities (vision, audition, gesture), which are human sensory facilities used to interact with our environment, and codes (language, sign language, pictorial language), which are systems of symbols, including their representation and associated constraints.

Approach

The general problem of storing "pure" or media-independent knowledge and, from it, generating rich and compelling interactive displays in a variety of forms is unsolved. We begin with a framework within which this problem can be attacked and partial solutions exploited to develop interesting experimental systems. We present an overall logical framework that can lead to a concrete software architecture; and a sequence of feasible research



There can be many applications, all communicating through the Knowledge Base.

Figure 1. Block Diagram of framework

steps toward systems that could realize our goal. We will attack the general, unsolved problem by developing several partial solutions within a framework that can lead to a more general solution.

Knowledge Representation

Some forms of knowledge can indeed be stored in a "pure" or media-independent way, and rich interactive displays can be generated automatically from this stored knowledge. This might take the form of logical propositions, which could be output in graphical, text, or other media to express the same basic knowledge. It might also take the form of tabular data, which can be automatically translated into appropriate graphs or other visualizations (MacKinlay, 1986)

For other kinds of knowledge, simple propositions attached to an automatic translation process may not provide enough richness. For example, a description of how to perform a surgical procedure would benefit from carefully designed pictures or animated graphic clips, but we do not know how to generate such automatically from a set of propositions. Such media-specific information might be represented directly in the given medium—e.g., movies, sound recordings—which would be annotated with propositions so that the knowledge system could use it appropriately.

An Initial Framework

We begin with the framework in Figure 1. In the diagram, circles and ovals represent information and/or data, and rectangles represent programs that transform data. The **Knowledge Base** (KB) has many types of information, including, of course, propositions that are media-independent. Overlapping the KB is a variety of **Applications** that interact via the KB. One application shown is a database manager for a given database (DB). Another application shown is central to our paper, namely, that of a **User Interface** (UI).

Each application has its own **Interpreter** that translates between the KB and the application. For the DB application, the Interpreter (1) identifies KB propositions that are requests for information that might be contained in the database, (2) translates each into an appropriate DB command, and (3) takes the DB's response and translates it back into KB propositions.

The UI Interpreter (1) identifies KB propositions that either request that certain information be displayed to a given user or request that a certain user be allowed to provide certain inputs, (2) translate these requests into UI commands that utilize a range of media, and (3) translate responses from the UI into KB propositions—these responses are usually due to user input. This particular interpreter is explored in more detail below.

It is interesting to note a similarity between the UI application in Figure 1 and the standard model of user interface software, with a user interface management system (UIMS), seen in Figure 2. That approach also separates the interface-specific information (syntax) from the underlying application functionality (semantics) (Jacob, 1998, Olsen, 1992). In both cases, there is an underlying core of (media-independent) information or operations (**Knowledge Base**, **Application Functionality**) that might be expressed in various ways and a separate component (**Interpreter**, **UIMS**) that converts that information into a specific presentation or interface to the user. Implicit in both is the notion that the presentation or interface component might be changed, without having to modify the knowledge base or application component in

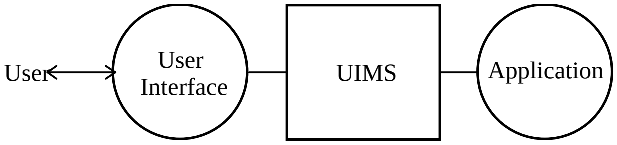


Figure 2. Standard UIMS architecture.

order to provide an alternate view or interface for the same knowledge or application functionality. While there has been some research in the user interface software area working toward systems that can automatically generate a user interface from a specification of the application functionality (Beshers, 1989, Foley, 1989), the framework itself (i.e., the dialogue independent application representation plus separate dialogue component) applies both to automatically-generated interfaces and manually-designed ones and holds for the research steps along the path from one to the other.

Refining the Framework

We now separate the **Interpreter** block for the User Interface application into three component parts in Figure 3: **Allocate Modes**, **Design** and **Plan/Execute**. The determination of precisely what information to present to the user and what inputs to allow the user to make is made by other applications and communicated via the KB. For the remainder of this paper, we assume this, and we explore further only the UI Interpreter.

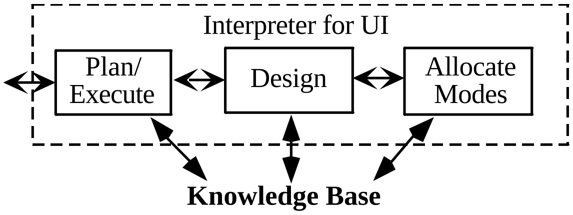


Figure 3. Refining the Interpreter for the UI

To realize the overall goal, processes must ultimately be provided to perform each of the tasks shown in the boxes in Figure 3. We attack the problem by seeking ways to approximate a final system by concentrating on some of the components and inserting partial approximations or manual steps for others, but always maintaining the same overall framework. Each of the steps is discussed further in the next section.

Filling in the Framework

KNOWLEDGE REPRESENTATION SCHEME

The **Knowledge Base** component contains information of various types. Central to our concern is "pure," media-independent information that can be presented to humans through a variety of different media. Other types of information include: media-specific information that is annotated with media-independent propositions (e.g., a sound recording of a speech annotated with propositions describing what it is, who spoke it, where, what was said, etc.); knowledge about how to translate various classes of information into various types of media presentations; knowledge about the human modalities and their connections to various types of media; knowledge about which information is currently being presented to the user and what media are being used for each.

If we follow this to its logical end point, we end up with the same problems faced by researchers of general natural language understanding systems. Not only do we need to represent knowledge about the world but also knowledge about media, about human modalities, and about the beliefs, desires and intentions of the "agents" involved. Fortunately, there are positions to investigate that are simplier than this eventual endpoint. We examine some of these points as they arise.

KNOWLEDGE BASE

As we said earlier, much of the knowledge in the KB comes from other applications and their respective interpreters. As we just discussed, there is also much knowledge needed to implement the **Interpreter** for the User Interface.

Initially, we will hand-code the knowledge in the KB, including the knowledge needed for the UI Interpreter as well as that which would "arrive" from other applications. For propositional knowledge where we do not yet have ways to generate good presentations automatically, we will, at least initially, allow the propositions to be augmented with any additional chunks of information needed to generate the presentations in various media, such as video clips, 3-D models, or narrative text. Along with this media-specific information, we will add propositional annotations, so that we can reason about them when deciding what to present.

ALLOCATE MEDIA

At this point, we must select media to use for the presentation we are about to generate. The choice may be constrained by the user's current situation (e.g., eyes-busy driving a car) or personal characteristics (e.g., visually impaired, dyslexic). Within those constraints, we then take the knowledge to be presented and decide what media to use and how to present it in those media. The most interesting cases will arise when information is best presented in a combination of media (e.g., a diagram with narration and highlights). Here, again, we can begin by performing this task manually. The operator simply tells the system explicitly what media to use for presentation of a given piece of knowledge. Again, we can later advance to an automated procedure while remaining consistent with the overall framework.

In automating this step, we hope to take advantage of recent advances in knowledge representation and reasoning for planning in AI. Levesque et al (Levesque, 1996, Scherl, 1997). use an extended situation calculus (McCarthy, 1969) to encode a logic that captures domain knowledge, knowledge about actions, knowledge about sensing, and knowledge about knowledge. While several such logics have been developed in AI, e.g. (Moore, 1985), the logic in (Scherl, 1997) admits some efficient implementations of planners, e.g. (Golden, 1994, Schmolze, 1998), that reason about knowledge. In addition to being able to reason about both the domain (e.g., how a machine operates) and about knowledge per se (e.g., the user already knows how part A operates), the tools for reasoning about sensing will allow us to reason about inputs (e.g., which inputs should the user be allowed to give and how should s/he encode them?).

We must also determine effective methods for representing and reasoning about classes of knowledge and how well they can be presented on various media. It may be helpful here to use communicative intent as a way to structure this task, organizing the classes of knowledge to be communicated by intent.

DESIGN

This task takes a given piece of knowledge and produces a presentation of it using the given medium or set of media. This task and the **ALLOCATE MEDIA** task lie at the heart of the problem and are the research areas that must be attacked first. They are discussed further below. The resulting presentation may be in the form of a high-level specification of constraints and goal to be satisfied. For example, it might contain timing constraints such as "Begin this narration before X but after Y." It will typically be underconstrained by these. A separate procedure will then find a solution to the constraints and produce a specific presentation that satisfies them.

PLAN/EXECUTE

In this step, we take the high-level specification just generated and reduce it to an actual presentation. This will typically require planning or scheduling tools. This could be included within the **DESIGN** task, but we segregate it because algorithms for performing it are a separate area of research, and some solutions already exist in this domain. For a starting point, we would simply incorporate known planning tools or algorithms to perform this task.

A Starting Point

How might research toward our ultimate goal proceed? We begin with the basic framework described thus far. As discussed, some of the individual tasks might be handled manually or semi-automatically at first in order to create initial prototypes, but the basic framework can remain intact through the transition to a fully automated process (unless it proves deficient). The first step would then be to implement this framework as a top-level software architecture and provide communication paths between the putative modules. Then each of the blocks can be filled in as described, at first with manual placeholders and/or with ad-hoc rather than general procedures. A future step would be to work on new ways of interacting with the resulting multimodal presentations, a new set of "multimodal widgets."

We consider here some limited or ad-hoc ways in which initial implementations for the key module, **DESIGN**, might be begun. Suppose the knowledge representation actually contains many kinds of representations within it. It will contain some high-level "pure" knowledge in the form of logical propositions. Other pieces of knowledge may be stored simply by saving several specific representations of them, one for each modality. They might be stored directly in the desired modality or in some intermediate form that is nevertheless tailored toward producing output in particular modalities.

Then we imagine the cross product of all possible translators between these various pure and impure representations and the various desired output presentations. Some members of this cross product may be impractical or produce poor outputs, so let us consider only a subset of this cross product. For example: For representation *A*, we have translations into modalities *M1* and *M2*. For representation *B*, we only have a translator into *M3*. For knowledge type *C*, it is already essentially in mode *M4*, so can only "translate" to *M4*.

Now given a piece of knowledge, we can produce a representation in one or more modalities, but only those for which we have provided the translators; in this limited version, we cannot yet present arbitrary knowledge in an arbitrary mode. For example: For knowledge stored as propositions, we can represent it in any modality and can provide a translator for every modality. For knowledge stored as a picture, only certainly output modes will make sense, so only those translators would be provided. For knowledge stored as one-dimensional time series data, there might be several widgets or codings for representing it in several different modes, and in some modes, several different codings for the same mode (e.g., alternate graphical plots). This suggests an initial, partial but highly feasible approach that can lead to an experimental prototype system as a basis for further research toward our ultimate goal of a more general system.

Interaction

The full potential of this approach is realized when we include interaction. Given a set of knowledge represented in a media-independent way, can we invent a palette of new media or ways of realizing and interacting with this information? The first step is to take the stored information and convert it to presentations that are designed for two-way interaction. This problem is analogous to information visualization or sonification in that it takes data in a pure form and creates new ways to display it. We may convert the pure information into different media as needed for different interactions. Supporting two-way interaction within our framework means that we can not only convert the pure information into various modalities but we can convert user input in various modalities back into our representation. This implies that we will need the converse of the **INTERPRET** process; we will need a two-way interpreter or converter or transducer operation.

Extending our framework to two-way interaction means that we can present information in one form for viewing and another for interaction if we wish. The user might interact with a three-dimensional plot of a set of data; the user's input will go back into the core knowledge representation; and the information as modified by the user's interaction may then be presented as spoken narration or some other form. We can provide different modes and multimodal combinations not only for presenting the underlying knowledge but also for interacting with it. An area for future research is the invention of new, possibly multimodal, widgets or controls for interacting with the modality-independent knowledge.

Conclusions

We began with the goal of storing the knowledge about how to repair a machine once, in some media-independent form, and then outputting—and interacting with—in different media and forms. We have now carved the problem into pieces but not yet solved any of them. We have proposed a framework within which this problem can be attacked and partial solutions exploited to develop interesting experimental systems. We presented a framework that can support a software architecture and a sequence of feasible research steps, and we have described some initial, prototypable components. The question we pose in this position paper is whether this architecture will serve well as a starting point on which to build or whether it has missing pieces or other deficiencies.

Acknowledgments

This work was supported in part by National Science Foundation Grant IRI-9625573, Office of Naval Research Grant N00014-95-1-1099, and Naval Research Laboratory Grant N00014-95-1-G014.

References

C.M. Beshers and S.K. Feiner, “Scope: Automated Generation of Graphical Interfaces,” *Proc. ACM UIST'89 Symposium on User Interface Software and Technology*, pp. 76-85, Addison-Wesley/ACM Press, Williamsburg, Va., 1989.

J. Foley, W.C. Kim, S. Kovacevic, and K. Murray, “Defining Interfaces at a High Level of Abstraction,” *IEEE Software*, vol. 6, no. 1, pp. 25-32, January 1989.

K. Golden, O. Etzioni, and D.Weld, “Omnipotence Without Omniscience: Efficient Sensor Management for Planning,” *AAAI'94 Conference on Artificial Intelligence*, pp. 1048-1054, AAAI, Seattle, WA, July 1994.

R.J.K. Jacob, “User Interfaces,” in *Encyclopedia of Computer Science, Fourth Edition*, ed. by D. Hemmendinger, A. Ralston, and E. Reilly, International Thomson Computer Press, 1998. (in press) http://www.eecs.tufts.edu/~jacob/papers/encycs.html [HTML]; http://www.eecs.tufts.edu/~jacob/papers/encycs.ps [Postscript].

H.J. Levesque, “What is Planning in the Presence of Sensing,” *AAAI'96 Conference on Artificial Intelligence*, AAAI, Portland, Oregon, July 1996.

J. MacKinlay, “Automating the Design of Graphical Presentations of Relational Information,” *ACM Transactions on Graphics*, vol. 5, no. 2, pp. 110-141, 1986.

M.T. Maybury and W. Wahlster, *Readings in Intelligent User Interfaces*, Morgan Kaufmann, San Mateo, Calif., 1998.

J. McCarthy and P.J. Hayes, “Some Philosophical Problems from the Standpoint of Artificial Intelligence,” in *Machine intelligence 4*, ed. by B. Meltzer and D. Michie, American Elsevier, New York, 1969.

R.C. Moore, “A Formal Theory of Knowledge and Action,” in *Formal Theories of the Commonsense World*, ed. by J.R. Hobbs and R.C. Moore, Ablex, 1985.

D.R. Olsen, *User Interface Management Systems: Models and Algorithms*, Morgan Kaufmann, San Mateo, Calif., 1992.

R.B. Scherl and H.J. Levesque, “The Frame Problem and Knowledge-Producing Actions,” Submitted to Artificial Intelligence, 1997.

J.G. Schmolze and T. Babaian, “Planning with Incomplete Knowledge and with Quantified Information,” Submitted to workshop on Interactive and Collaborative Planning, in conjunction with Fourth International Conference on Artificial Intelligence Planning Systems (AIPS-98), 1998.