Reminders of ToDos-II (R2Do2): Work Flow Agents for Reengineering Health Care Enterprises

by

Barry G. Silverman, Christo Andonyadis, Alfredo Morales, In-Kuk Song
Institute for Artificial Intelligence, George Washington University, Washington DC 20052
barry@seas.gwu.edu

Abstract

This paper describes efforts to develop and field an adaptive health planner and parallel agents that securely connect distributed users and data sets in a collaborative approach able to anticipate health ToDo items and to remind and alert about these items over the web. The planner and agents, reasoning temporally and non-monotonically, are able to handle changing health conditions, unknown health facts, user wellness and lifestyle preferences, and updates about each user's situations. A set of robust, open standards-based client server approaches (e.g., CORBA, ODBC, Java, and ANSI HL7 Common Object Model) are used to encapsulate and plug all these components into the web, and to handle the distributed agent and object communications. Extensions to these emerging standards are at times made due to engineering and domain considerations, however, the goal is to maximize reuse of published guidelines and integration of R2Do2 by any patient record system that also adheres to the standards. From this perspective, R2Do2 is an experiment in an open standards framework for middleware in the healthcare field. This research also tries to reflect lessons learned about the extensions needed in these standards if healthcare middleware frameworks are to transparently support users over the web.

1) Introduction

Several related paradigm shifts are occurring in the health care field to produce an opportunity for a dramatically more cost-effective delivery system. First, medical practice is shifting toward more prevention, health promotion, shared patient-provider decision making, and managed-care. This shift is creating a greater demand for online services and information. Second, distributed, heterogeneous patient data and isolated applications are becoming less tolerated in view of progress in health informatics standards, the information highway, and a proliferating set of access devices. This implies health system users want a shift toward usable interfaces, and easily accessed, integratable information sets. Third, due to cost and quality control concerns, inspection, rework, and paper intensive processes are giving way to reusable electronic information sets that are machine-checked for omissions and to get the decisions right the first time. Lastly, to meet these demands, applications developers are attempting to shift formerly closed, proprietary applications toward open, rapidly customizable software. This, in turn, will further democratize health information delivery.

To enable these types of paradigm shifts, one would like to utilize the Internet and the World Wide Web ("the web"), since it is widespread, supports cross platform interconnection, promotes heterogeneous information systems, and has an installed base of open standards based software browsers and distributed servers. Yet, the web alone suffers several drawbacks. Among others, the current obstacles to using the web include:

1) It is largely document-centric and ignores the relational technology that is so central to much of the patient record and personal health information systems of the medical sector. It is important to explore how best to utilize plugins, object resource brokers, and other open database interfaces to integrate the document- and data-centric worlds,

2) The web is largely a browse and surf environment in which users search for information and then "pull" it out. An alternative is for the web to proactively anticipate information needs of the users and to "push" it toward them. This is the agent approach, in general. In the health care field, an important type of proactive agent is computer generated reminders and alerts. The advent of the Java programming language and the inclusion of applets in the hypertext markup language opens up the prospects of agent push approaches. An open question is how can the methods of artificial intelligence, software agents, and Java be scaled to the needs of large communities of health care users.

3) The web, being a document-centric environment, is often devoid of active social presence. Yet both consumers and providers require a high degree of human contact for health care services, consults, support groups, and the like. This is beginning to shift, particularly as web-enabled browsers begin to extend support for cooperative environments such as email, graphical bulletin boards, electronic white boards, chat groups, and interactive video. This is a broad topic, and this paper will only briefly examine it from the view of how agent push approaches can be integrated.

4) The web and the Internet grew to prominence largely on their openness and how they fostered democratic values, freedom to roam, and barrier elimination. Yet health care involves sensitive data, private information, and "firewalls" that assure greater security and confidentiality.

The latter must be incorporated while sacrificing as little of the former as possible.

Given the paradigm shifts in health care, progress is needed to reduce each of these four sets of obstacles. At present they present a collective challenge to any health-related project, and this article reviews how one project is trying to address them with particular emphasis on 1 and 2. Specifically, this paper describes a reusable intelligent agent project that seeks to unify the data- and document-centric views and to provide a push approach in a collaborative and secure environment. This agent approach is being built with open standards-oriented technology that is maturing at present, but is expected to be widely implemented in the near future.

The system we are developing, called Reminders of ToDos-II (R2Do2), offers a model for how to meet the four challenges in the area of health care planning in general, and for reminders and alerts, in particular. Medical reminder and alert systems organize and display machine-triggered ToDo items, reminders, and alerts. Studies show these machine generated reminders and alerts significantly increase performance of clinicians as well as consumers/patients, and if widely deployed they hold the potential to save the health industry billions of dollars annually through better disease/error prevention and health promotion (Shea et al. 1996) (Einarson 1993). In the health care arena, this approach also holds the ability to empower consumers to better manage their own wellness efforts, and to enable clinicians to reduce omissions and unintended medical errors: Safran & Rind (1996). Further, it should enhance communications and collaboration between health care consumers and providers. More generally, this paper illustrates an approach to adaptive planning for personal assistant agents.

This article reviews these innovations and identifies how R2Do2 serves as a model for further unification efforts. Specifically, Section 2 presents the architectural view of R2Do2 and shows how the four web-related challenges are addressed for both institutional and personal versions of R2Do2. This raises a number of challenges for scaling the AI algorithms and agents and for incorporating numerous open software and data standards as are discussed in the next two sections of the article, including the adaptive planner (Section 3), and the temporal reasoning, reminder agents (Section 4). Finally, Section 5 returns to the topics of the Introduction with lessons learned and future research needs.

1.1) Healthcare Guideline Agents

The idea is not new of online alerting and reminding from electronic practice guidelines able to interact with a medical record and with providers' orders. Early examples may be found in McDonald (1976) and Warner (1972), while more current examples are described in Musen et al. (1996), Safran & Rind (1996), Shea et al. (1996), and many others. However, for a variety of reasons, very few of these systems have made it out of the academic center, hospital, or clinic where their developers have piloted and tested them.

The major obstacles to portability seem to be severalfold: (1) the reminder methods and engines are not generic, but are written in a variety of languages and according to arbitrary, undocumented standards so that the code is difficult to modify and maintain; (2) the reminder methods and engines are hard-wired to "legacy" or institution-specific record systems with local lexicons and interface code; (3) the user or screen interfaces are similarly hard-wired and non-standard; (4) the guideline KBs are specific to the home institution that authored them and may not be desired by others without modification; and (5) the programmers who created the code are frequently unavailable to modify it later.

Some vendors offer new, turnkey patient record systems that try to overcome such obstacles. For example, Healthpoint ACS (anon. 1996) is a patient record system for small private practices that includes an integrated module that scans all prescription orders against other drugs on the patient's medication list as well as against the patient allergy list. Right out of the box, its conditions rulebase reminds the user when contra-indications and adverse interaction effects will arise. One can add to or edit these rulebases to make them institution-appropriate. Unfortunately, such vendor-specific systems are a necessary, but not sufficient, step to overcome the obstacles. This is because most information systems already exist, and they won't easily be converted to a given vendor's implementation, nor do their institutions wish to do so.

Another answer is to connect applications to middleware so that others may view the application as a component that they may embed and or plug in by also connecting to the middleware. One manual approach to this is where component or object vendors have taken to "wrapping" legacy information systems and adding their applications via integration interfaces. An example is Multum Inc's MediSource expert system for drug therapy and order-entry (Schrier 1994). Originally a stand-alone expert system, MediSource now offers a consulting service that manually embeds their software as an agent obtaining order entries from the legacy system records, and submitting its alerts to the clinician through the familiar legacy interface. This is a solution that could be duplicated with practice guideline knowledge bases (rather than just drug therapy), however, it fails the test of open standards-based software, and is once again vendor-dependent. This solution does not scale well or quickly when trying to expand to each new site in a given healthcare community, when adding other potential knowledge bases or agents, or when extending over the distributed, longitudinal, heterogeneous "record."

The alternative pursued by R2Do2 is to build towards a middleware architecture, called HOLON (Silverman et al., 1996), that adopts interoperability and open-software standards. R2Do2 is a test application of the HOLON project meaning both R2Do2 and HOLON are being evolved in parallel. HOLON uses a three tier middleware architecture. This idea is an "any-any" model to connect any client to any database system via an open-standards middleware component. An editable/extendable reminder/alert agent such as R2Do2 can plug into this middle tier. Specifically, to integrate guideline agent capabilities (or other applications) with legacy systems via a three tier approach, HOLON is using:

- a semantically rich, open, object-oriented, standard for specifying and accessing heterogeneous, longitudinal, distributed information (It is not intended to provide new information, nor to replace existing information standards as far as they go, only to adopt an open, standards-based meta-model where others can plug in their datasets and applications. To do this we explore the HL7 messaging standard as an object oriented model of the clinical data repository, and extend it at the information content level via use of ICD 9 and 10 codes, among many other content standards),
- the wrapper, lexicon translation, vocabulary server, and integration engine capabilities to query/manipulate and plug-together new and legacy systems,
- traditional middleware services such as communications, transport, security, message queuing, etc., as well as emerging standards for application mobility and object brokering (ew.g., CORBA, OLE), and

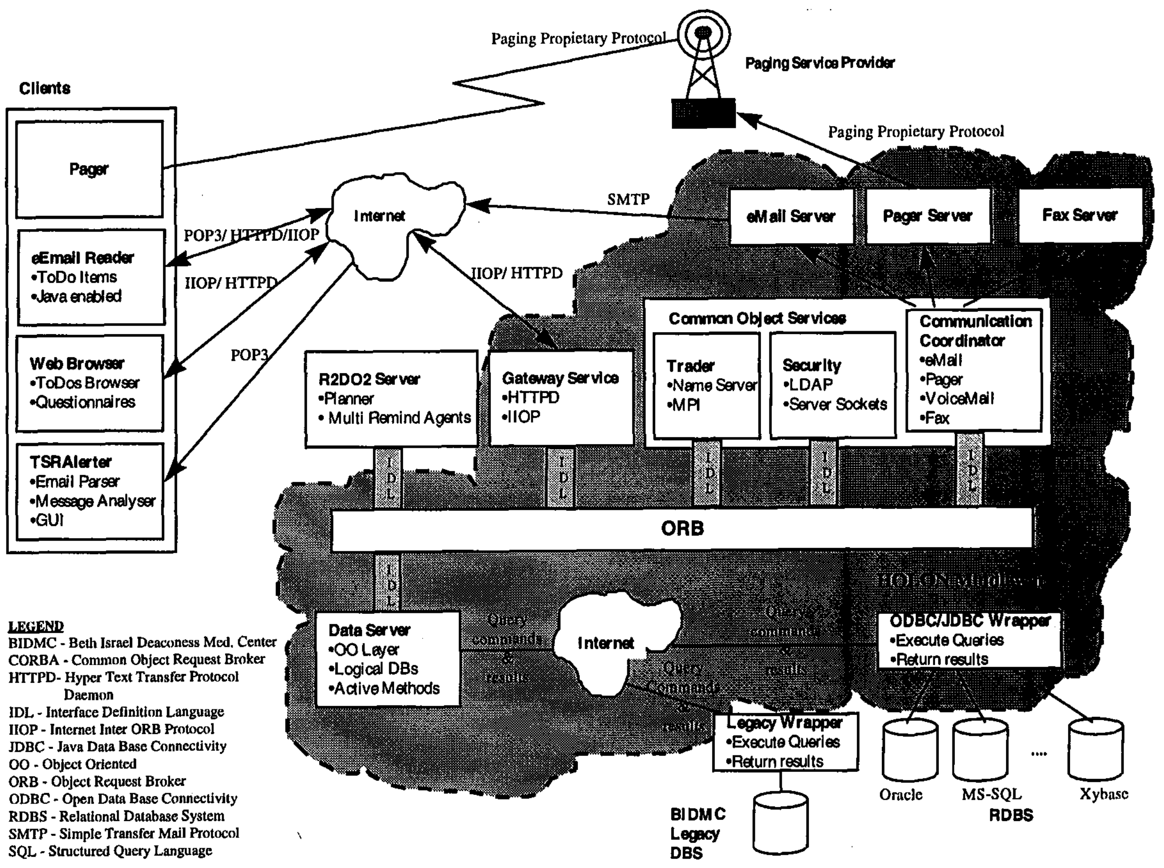
The "middleware" field partially addresses many of these items: (Bernstein 1996), (Lewis 1995). Yet sector- and industry-specific frameworks sitting above traditional middleware are also being developed to complete this list and to open up the full agent potential. In the health care sector, a number of standards bodies and consortia are working in these directions: (Fitzmaurice 1994), (Halloran et al. 1996), and (Rischel 1996). Our research investigates how a reminder engine can exploit some of these advances. In this way R2Do2, sitting atop HOLON, is a model of a middleware framework for the healthcare field (i.e., a vertical facility in CORBA terms).

This is a point also made by Musen et al.(1996) in their EON system. EON is an environment for authoring electronic protocols and connecting them to patient record systems so as to automate the recommending of therapeutic interventions. Musen et al. repeat the point that cross-industry middleware frameworks like CORBA and OLE are vital advances. Yet "for clinical software to be reusable and maintainable in a proficient manner, higher level software abstractions are necessary..." to promote the clinical-specific middleware field. R2Do2 shares that objective of a more healthcare oriented middleware framework with EON, however, there appear to be several differences between the two systems: (1) EON seems to be oriented toward supporting a user at a clinical workstation, whereas R2Do2 is concerned with supporting a distributed group of users; (2) EON seems to focus on clinical recommendations, whereas R2DO2 covers both the clinical and consumer perspectives; (3) EON offers fine-grained recommendations for both the diagnostic and care tasks, whereas R2Do2 to date provides only coarse-grained support for prevention and wellness promotion tasks; (4) there are internal algorithmic differences for accomplishing roughly equivalent clinical tasks such as the heuristic propose-revise-plan algorithm of EON as compared to the nonmonotonic, adaptive argumentation logic of R2Do2; and (5) EON provides a robust framework that middleware standards need to be connected to, whereas R2Do2 is currently attempting to conform with and utilize a number of middleware standards such as CORBA, HTTP, IIOP, Arden, HL7, and the like. These differences reflect the fact that no single framework is sufficient to address the entire field of middleware for healthcare, and that multiple frameworks should be encouraged. However, it is important that all the healthcare middleware frameworks be integratable if application users are to make use of them all.

2) The R2Do2 System

The R2Do2 arhitecture makes use of several principles of modern software engineering such as object-oriented design (in its modules, communications, and databases), open software standards, portability of interfaces, and three tier client-server configuration. In Figure 1 one can see multiple clients on the left, the servers on the right (and bottom), and middleware connecting the two. Before we discuss the details of Figure 1, several items are worth pointing out. First, this architecture avoids the "stovepipe effect" or closed, proprietary, non-portable interfaces and databases found in much legacy software in the health care field. By contrast, the architecture of Figure 1 permits virtually any vendor's hardware to be seamlessly substituted for either the client or server stations. R2Do2 is platform independent. Likewise, any legacy patient record system is potentially connectable once the wrapping is effected. In the same vein, browsers, collaborative environments, personal information managers, and the like can be readily substituted just by registering them with the common object services. Finally, the human notification process is not limited to computer clients, but is extensible to pagers, interactive TV, fax, voice synthesis, and other forms of reminding and alerting.

Figure 1 - Architecture of R2Do2 Showing Its Use of HOLON's Three Tier "Client-Middleware-Content Server" Model



LEGEND
BIDMC - Beth Israel Deaconess Med. Center
CORBA - Common Object Request Broker
HTTPD- Hyper Text Transfer Protocol Daemon
IDL - Interface Definition Language
IIOP - Internet Inter ORB Protocol
JDBC - Java Data Base Connectivity
OO - Object Oriented
ORB - Object Request Broker
ODBC - Open Data Base Connectivity
RDBS - Relational Database System
SMTP - Simple Transfer Mail Protocol
SQL - Structured Query Language

Returning to the details of Figure 1, there are six major modules that warrant discussion: clients, groupware, middleware ORBs, middleware services, reminder server, and data server. Perhaps more important than any module, though, are the users who consist of healthcare consumers and providers, each of whom have differing needs from the modules. We will mention these range of needs and how they are supported in the discussion of the modules.

Clients -- The initial client design target is a desktop PC, although our consortium also requires interactive TV and beeper service. The client includes (see Figure 2) a web browser, web-enabled email viewer, and a terminal stay-resident (TSR) alerter component that parses email and alerts users when high priority health items have arrived. The web browser is the main route through which applets elicit "user preferences" for health plans, recurring ToDo items, and reminder schemes, and through which the database is browsed. The email viewer is the primary place for receiving and perusing ToDo item reminders and for altering their settings, forwarding them to others, and so on. All three elements of the client make use of the Java virtual machine for executing applet bytecodes, and thereby maximize platform independence, multi-threading, and security of local resources.

Groupware -- R2Do2 explicitly incorporates an email router with a forward and CC list for relevant consumers, providers, and surrogate caregivers. Also, R2Do2 can be incorporated into any open standard groupware environment, such as a web browser that recognizes CORBA and Java applets. Thus it can be used alongside whiteboards, chat spaces, and video-conferencing facilities as in a test-bed currently being deployed at numerous homes, clinics, Beth Israel-Deaconess Medical Center in Boston and Norwalk Hospital, Connecticut (e.g., see Silverman 1996).

Middleware ORBs -- Rather than connecting to servers, the clients connect instead to open standards middleware, or more specifically via HTTP and the Internet to Object Resource Brokers (ORBs). The ORBs effectively encapsulate the various client and server modules, and turn them into object-like components that can message each other transparently, through a common object model. At present, R2Do2 subscribes to the model of the Common Object Request Broker Architecture (CORBA), an industry sponsored standard (e.g., see Otte et al., 1996). We use the ORB from Visigenic, Inc. a subsidiary of Netscape Corp.

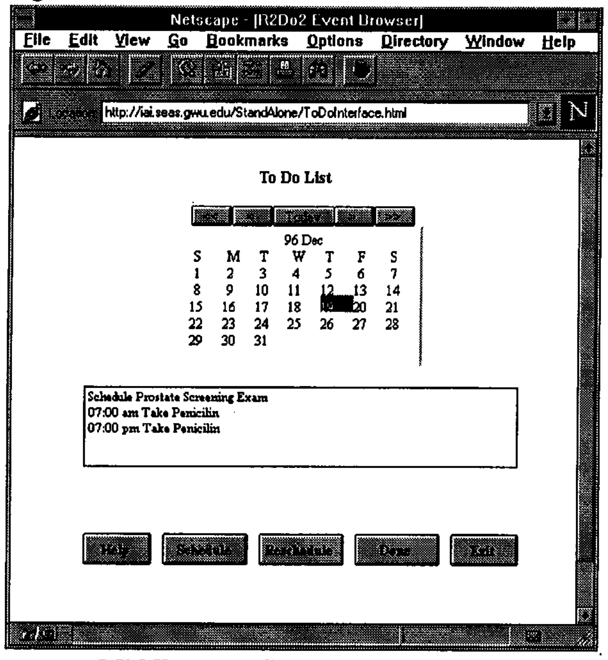


Figure 2 - R2Do2 Event Browser Screen

Netscape - [R2Do2 Event Browser]
File Edit View Go Bookmarks Options Directory Window Help
N

http://iai.seas.gwu.edu/StandAlone/ToDoInterface.html

To Do List

```
            96 Dec
 S   M   T   W   T   F   S
 1   2   3   4   5   6   7
 8   9  10  11  12  13  14
15  16  17  18  19  20  21
22  23  24  25  26  27  28
29  30  31
```

Schedule Prostate Screening Exam
07:00 am Take Penicilin
07:00 pm Take Penicilin

Middleware Services -- Along with the CORBA standard comes a set of run-time library-like services. New users are registered with these services to provide Master Patient-Member Indexes; security authorization, access, and authentication information; and notification alternatives. In the personal version of R2Do2, many of these services are omitted, while those that remain are automated. In the institutional version, this activity is partially supported by applets collecting data from users, but is highly dependent on a system administrator to verify, update, and maintain the various registrations.

Data Server -- This server contains computerized patient records, personal health information not normally in patient records, and PIM type activity information generated by the consumer from their screen. The persistent store is, in effect, a dynamic model of the consumer's health status and activities that is jointly maintained by the consumer and by health care providers. We have deployed an object layer sitting at the top of the persistent store that monitors the input streams, detects trigger conditions (e.g., users whose records have been updated since the last broadcast), and transmits these items to the planning component and reactive agents.

In practice, connecting institutions to R2Do2 is a non-trivial task due to cross-institution guideline differences, lexical and semantic ambiguities in diverse patient record terminologies, and legacy storage system wrapping and translating constraints. We are currently in the second of a three year project in which a non-profit consortium is piloting more generic ways to connect and wrap such institutional resources including use of the ANSI healthcare data model, open database connect (ODBC) standard, and the UMLS vocabulary server: Silverman (1996).

Reminder Server -- The classical definition of a planner is a program that generates a sequence of actions (the plan) that will move an entity from its initial to its goal state. A plan is first generated and then sent to an execution agent to carry out each of the (ToDo) actions. A number of researchers have extended this approach so it responds to a dynamic, uncertain environment where the initial plan may need to be revised before execution is completed: e.g., see Haddawy(1996), Lyons & Hendriks (1992), among others. These approaches differ in terms of, among other things, varying degrees of world knowledge in their planner's model of the environment, sophistication of adaptive reasoning, and degree of separation of sensor, planner and effector. The best way to handle these and other differences are open questions.

In our case R2Do2 adopts a framework for organizing the distributed reasoning and processing elements. The framework settled on here has 5 steps that R2Do2 continually strives to solve. The steps push the system to:

1.maximize consumer health by recommending prevention and care plans for each situation that arises

2.mine the consumer health files with the help of a Java version of the CLIPS forward chaining engine and a set of guidelines KBs in order to discover situations requiring action and to construct the plan of recurring ToDo items for each consumer

3.map the recurring plan elements (ToDo items) onto the consumer's (and provider's) schedule and launch agents to help carry out the scheduled plans, subject to user wishes.

4.utilize personal assistant agents to isolate and carry out a user-acceptable scheme to notify and remind the consumers of the ToDo items likely to maximize their health

5.alert providers and surrogate caregivers when consumers seriously deviate from the plan to their self-detriment.

In R2Do2's framework (see Figure 1), the Planner generates a number of ToDo items for each consumer and provider registered in the environment. Numerous plan intervals and actions may need to be generated at various times during the year for each user. Once new or updated plans are completed, the agents are launched and are given both personal assistance and limited self-planning capabilities. These capabilities allow the agents to react directly with users and with the client servers. Thus agents can conserve resources by putting themselves to sleep until needed; alter their reminding schemes in response to user preference; postpone, cancel, or re-assign ToDo items and reminders also upon user request; coordinate alternate modes of communication (e.g., TSR alerter, email, and beeper); and contact providers and inform surrogate care givers when critical ToDo items are neglected.

Via the "user request" mode, the users may alter their plan directly (e.g., snooze, cancel, or reassign an item). Also, users can add new ToDo items either directly through applet screens, or indirectly via their answers to health profile and preference questionnaires.

In this manner R2Do2 derives an initial health plan that it continually refines and adapts as the users age, change their practices, develop new conditions, and so on. As new, or sometimes unexpected events occur (e.g., a miscarriage, a diabetic complication, reaching the next birthday, etc.) the original plans will need to be revised anew, and the agents carrying them out, modified yet again.

3) Empirical Findings

Over the past year, we have built several prototypes and two versions of R2Do2. At present we have both a personal version available through a web browser interface, and an institutional version being piloted at the Beth Israel-Deaconness Medical Center. We explore the results to date and lessons learned in the subsections that follow.

4) Lessons Learned

This paper described efforts to develop and field an adaptive health planner and parallel agents that securely connect distributed users and data sets in a collaborative approach able to anticipate health ToDo items and to remind and alert about these items over the web. The planner and agents, reasoning temporally and non-monotonically, are able to handle changing health conditions, unknown health facts, user wellness and lifestyle preferences, and updates about each user's situations. Usage of and extensions to argumentation, qualitative decision making, situational calculus, and temporal algebras are described. Defeasible assertions, default arguments, support sign dictionaries, closed world assumptions, and anytime algorithms are used to derive feasible health plans that are later updated as more information unfolds. A set of robust, open standards-based client server approaches (e.g., HOLON, CORBA, ODBC, Java, and ANSI HL7 Common Object Model) are used to encapsulate and plug all these components into the web, and to handle the distributed agent and object communications. Extensions to these emerging standards are at times made due to engineering and domain considerations, however, the goal is to maximize reuse of the guidelines by any patient record system that adheres to the standards.

At the start of this paper, four challenges to web-based applications were introduced that we return to in the next four subsections.

4.1) Merging Data with Document-Centric Computing

Web pages with their hyper-text markup language (HTML) are a document-centric mode of computing that has revolutionized usage of computers. In R2Do2, guideline KBs, questionnaires, preference elicitation screens, and ToDo item messages and advice utilize this document modality. In trying to extend this approach to the data-centric side of health information management systems, several difficulties arise that are not easily overcome. These difficulties center around the collection, storage, retrieval, and maintenance of each user's health information and situations, what is earlier defined as Fi. For a new user, the Fi is a large set of questions to answer. This is often called the health profile or patient interview, and it may involve collecting hundreds of answers. Ideally, one would like to use a question engine that can branch over interview rules and collect those answers most relevant to a given user's health situation. Unfortunately, the web standard inhibits reaching such an ideal if one seeks to adhere to HTML standards such as cgi forms and Java virtual machines. For example, one cannot download the Java inference engine and run it within an applet since local memory resources cannot be written to. Waiting for the engine to read and write assertions on a remote client is impractical. Likewise downloading a plugin interview system can be discouraging to users. The simplest alternative is to use fixed questionnaires about each health area and situation, and to stream these to the user as cgi forms or Java applets. This means that Fi are collected in a more exhausting, less user-sensitive manner than one would like. As a result, critical Fi may remain unanswered longer than would otherwise be necessary.

At the start of this paper, several paradigm shifts in the health care field were mentioned, one of which was the idea of empowering users by connecting them to access and manage their own health data. Using the web browser, Internet Inter-ORB Protocol (IIOP), an ORB bus, and the ODBC standard we have demonstrated that this is feasible, as have others before us through comparably open standards (e.g., as did Kohane et al., 1996 with HTTP). However, there are numerous, serious problems to this approach. Here we will concentrate on the connectivity problems, while subsequent paragraphs will address programming and security aspects. The connectivity problems straddle the syntactic and semantic levels. At the syntactic level, there is the basic problem of connecting the "pipes." The main obstacle to point out here is that the CORBA standard has not yet matured to the point needed. In particular, the ORB services include, among others, a trader, a name server and a database query server. The name server standard, however, is still underspecified so that it is not possible to locate a given patient and their records -- that is being addressed as of this writing by request for proposals for a master patient index standard by the CORBAmed committee. The query server standard within CORBA was adopted in the summer of 1996, but that standard in turn adheres to the SQL3 and Object Query Language (OQL) standards that are not yet widely available in current database products. For these reasons, we found we had to implement our own name and query servers. In the former case we are depending on a HOLON consortium partner (Concept Five), while in the latter case we replaced the OMG Query Server with a Java ODBC-standard based connection outside the ORB. That is why earlier Figure 1 shows the distributed databases in our current testbed connecting through the data server, rather than directly to the ORB.

At the information content level, there are more obstacles to the connectivity problem. Our goal is to utilize site-independent software and lexicons via use of common object standards such as ANSI's HL7, among others. With this approach, each institution that wishes to connect its legacy systems to R2Do2 must go through a translation and wrapping effort as is currently ongoing at our Beth Israel-Deaconess and Norwalk Hospitals testbed sites. Once they conform to the accepted standards, they can use the reminder engine since R2Do2 attempts to restrict its guideline lexicon and patient data references to the same accepted vocabulary standards. However, the vocabulary sets that need to be referenced are not under the purview of any single standard setting body. There are a variety of authoritative or de facto standards such as, among others, ICD-9 diagnostic codes, LOINC laboratory order codes, Observation Ids, and so on that are not part yet of the HL7 standard. We are trying to stay with the more well established of these coding schemes as embellishments where HL7 is underspecified. The ultimate settling of these vocabularies is a challenge the standards groups need to settle for this approach to work. Another challenge for standards setting is that HL7 tables do not currently exist that support hierarchical inferencing, such as , for example, "has the patient taken any type of pertussis vaccine?" At present one must manage all such processing within R2Do2. It would be useful if the standards would recognize decision support needs as well as information retrieval.

A final issue concerns the Arden Syntax standard for medical logic modules. Our guidelines KBs conform to this documentation standard in an effort to promote their reusability and local modification. A future version of R2Do2 will also be providing an HTML viewer of the guideline KBs in Arden form. We also would like it to permit English level editing of the KBs. One major obstacle to this idea is the Arden Syntax's inability to standardize how a medical logic module gets access to data. It needs both events and the ability to retrieve from the patient's "record" or from the data repository. The Arden Syntax Standard leaves the precise access syntax un-standardized, with the idea that it would be written locally because of the different IS environments. This inhibits portability and reuse of logic modules since the local rewriting of that section is time consuming, error prone, and requires person to person consultation between someone at the author's institution and the person attempting to reuse it. An alternative we are trying to provide for this section is to embed a full set of CORBA interfaces directly connected to HL7 and other (finer grained) standard code "events" (e.g., 'a new DT booster was administered') pushed to it by the object layer of the medical record data repository (Fi). To date, our approach seems feasible and we think it would prove a viable way to extend the Arden Standard. It would also allow the creation of structured-English level editors of the guideline KBs and the various rule elements (events, actions, etc.) in which users could avoid the need to code access connections or to deviate from approved repository vocabulary.

4.2) Integrating Personal Assistant Agents

Solving the AI side of a class of applications, such as R2Do2 does, is no easy task. It involves working out theories, algorithms, engines, typical planning problems (qualification, ramification, frame, closed world, defaults, etc.) and KBs. At the end of such an effort, the typical AI person expects to be done with the project. However, as this article amply illustrates, integrating the AI into the Internet and the web adds an entire new layer of issues and effort. The costs of that level included about a person-year-equivalent of added effort, above and beyond the AI- and KB-related effort. The advantage of this effort is that as soon as it is done, the AI environment is immediately available to the world.

A related issue is that no matter how much effort one puts out, there are some obstacles that can't be eliminated. With enough effort, one can overcome CORBA and IDL learning curves, database connectivity obstacles and so on, but some aspects of the agent-web integration are immovable. For example, in the field of reminding, the PIMs have set a standard for the metaphors and screen look and feel. Our screen widgets to date are restricted to those shown in Figures 2 and 6. Without resorting to PIM plugins, the screen interfaces are relatively plain. While this will change in the near future, as of this writing, we have few widgets for calendar presentation, day and weekly planners, alarm bells, clocks, recurrence sliders, and so on. Using PIM plugins would solve this dilemma, but would violate the web ideals of easy portability, rapid streaming, and local resource security. Also, installing the plugin on low end machines, those with low baud rate modems and limited extra memory, might be unappealing to the users who own them.

Whether embedding the R2Do2 planner and agents in PIMs or directly in the web, this is a good place to bring up an important issue we have been grappling with throughout this article -- that of reliability of the agent and its advice. There are two levels of reliability of concern here, logical and engineering reliability. At the logical level, R2Do2 uses a mathematical framework for organizing its distributed processing and temporal reasoning. The discussion at numerous points has attempted to draw parallels between this framework and the more traditional frameworks of the nonmonotonic logic literature. It would be interesting to explore a more formal framework of logic for R2Do2, one that addresses satisfiability and correctness proofs of the HealthPlan. However, in the interest of practicality, R2Do2 did not start with such a rigorous goal. There are numerous concerns that such a framework must overcome, many of which have been mentioned such as the frame and qualification problems, the lack of a complete KB of rules to defeat all defaults, the computational difficulty of satisfiability tests required in proof procedures, and others. Nevertheless, a more formal approach or alternate formalisms may offer numerous benefits (maximizing health, assuring cost objectives, satisfying user preferences, etc.) and they warrant further research.

From the engineering reliability perspective, no matter how reliable a plan or plan defeater may be, it has no value if it does not reach its user properly. This brings up a number of issues we have so far ignored about Internet communication failures. At present, we have not ensured recurring reminders cover all the scenarios for communication failure that may arise. One of the reasons CORBA works is that it is essentially stateless. It may be that our planner and agents are too stateless as well, particularly in terms of tracking the outcome of their conversations with one another. What happens if R2Do2 launches an Mk that is not received by the agent or sends a health alert email about a consumer that is never received by the caregiver? These and other engineering reliability issues have not yet received sufficient attention that we can adequately answer such questions. If R2Do2 is to graduate from a web curiosity to a serious tool for institutions and individual consumers, we will need to conduct extensive engineering studies and reliability hardening efforts.

4.3) Web-Interactive Social Presence

One can get numerous reminder and alert benefits from a personal version of R2Do2 that doesn't need to connect to anyone else. Provided the user inputs their health profile, Fi, the guideline KBs will fire and generate ToDo items and plans. Also, R2Do2 offers useful templates and default reminder schemes for user-specified ToDo items such as pills, recurring shots, a few areas of chronic care, and the like. All that is needed to get these features is a Java-enabled web browser.

Yet the institutional version of R2Do2 seeks to extend these features. When R2Do2 is connected to institutional databases and patient record systems, it can give the caregivers reminding and alerting messages about patients coming in for an appointment that day, patients needing appointments, and consumers who have situations requiring attention. Consumers in turn get connectivity to their caregivers both via email reminders, and for chronic care triggers. For this connectivity to happen, in addition to a Java- and email-enabled web browser, each user must install a terminal stay resident alerter and email parser on their machine. This is a Java application. Since they are installing these base items, one could bundle this with other collaboration features useful for a remote care environment, such as desktop video conferencing, electronic whiteboards, and support group chat space. Our current testbed for the HOLON consortium includes such items. Results to date show they do not increase the complexity or effort for the R2Do2 developers.

4.4) Security and Privacy on the Web

Although these are critical issues, our approach to security and privacy thus far has been purposefully low key. For the personal version of R2Do2, security and privacy would be preserved by the consumer if they could install the entire system on their local machine. Thus one could price a commercial version that would maintain security and privacy relatively easily. This would be unaltered if we embed R2Do2 within a PIM.

If we maintain a central server for supporting personal usage, however, this perception of security and privacy is compromised. Most users would probably be reluctant to place personal data on a quasi-public server, even if we assured them our current version has been implemented within the Java virtual machine to prevent foreign code reading/writing on their local disk, with secure sockets during ORB transmissions, with access password-controlled by the name server, and with authentication and auditing provided by a HOLON security module: i.e., see Harris et al. (1997). Still, we have taken no further effort on the belief that the web in conjunction with vendors like Visigenic (ORB), Concept Five (HOLON Security Module), and others will ultimately produce a solution that is widely perceived as allaying security and privacy concerns. For institutional intranets that adopt firewalls and pursue vigorous security management procedures and policies, our current configuration may be close to what they will find acceptable. This latter issue is being explored by the Beth Israel-Deaconess Medical Center in the current HOLON testbed and will be the topic of a future report.

5) Concluding Comment

R2Do2 is an experiment in extending middleware in the healthcare field. This framework makes use of open standards for architecture, software, guideline KBs, clinical repository models, information encodings, and intelligent system modules and agents. By pursuing the use of such standards we hope eventually to maximize immediate reusability of the R2Do2 framework by others who also adhere to these open standards. This research also tries to reflect lessons learned about the extensions needed in these standards if healthcare middleware frameworks are to transparently support users over the web.

ACKNOWLEDGEMENT
The financial support of NIST/ATP, the HOLON Consortium, KOOP Foundation, and Beth Israel Hospital are gratefully acknowledged.

REFERENCES

Allen, J.F. 1984. Towards a General Theory of Action and Time. *Artificial Intelligence*, 23/123-54.

Anon. 1996. *Guide to Clinical Preventive Services (2nd Ed.)*, Washington, DC: US Public Health Service.

Anon.1. 1996. *Introducing Health Point ACS*. Durham, NC: HealthPoint Inc.

Bernstein, P.1996. Middleware: A Model for Distributed System Services. *CACM* 39(2):86-98.

Einarson, T. 1993. Drug Related Hospital Admissions. *The Annals of Pharmacology* 27:832-40.

Fitzmaurice, J. 1994. *Putting the Information Infrastructure to Work: Health Care and the NII.* Washington, DC: DHHS, AHCPR Pub. No. 94-0092.

Haddawy, P. 1996. A Logic of Time, Chance, and Action for Representing Plans. *Artificial Intelligence* 80:243-308.

Halloran, M., et al. 1996 White Paper: Accelerating the Movement Toward Standards-Based Interoperability in Health Care. New York: IEEE-USA Medical Technology Policy Committee (Clinical Information Systems Subcommittee).

Lewis, T.G. 1995, Where is Client/Server Software Headed? *Computer*. 4:49-55.

Lyons, D.M., Hendricks, A.J. 1992. A Practical Approach to Integrating Reaction and Deliberation. in J. Hendler (ed.), *Artificial Intelligence Planning Systems*, San Mateo: Morgan Kaufman. 153-162.

McDermott, D. 1982. A Temporal Logic for Reasoning about Processes and Plans. *Cognitive Science* 6:1-55.

McDonald, C. 1976. Protocol-Based Computer Reminders. *New England Journal of Medicine* 295:1579-81.

Otte, R., et al. 1996. *The Common Object Request Broker Architecture*, Englewood Cliffs: Prentice Hall.

Pollock, J.L. 1992. How to Reason Defeasibly. *Artificial Intelligence* 57:1-42.

Rishel, W. 1996. Software Components, the Clinical Workstation, and Healthcare Networks. in *Proceedings of the Healthcare Information Management Support Systems (HIMSS) Conference* 1-13.

Safran, C., et al. 1996. Effects of a Knowledge Based Electronic Patient Record on Adherence to Practice Guidelines. *MD Computing* 13(1): 55-63.

Schrier, R. 1994. *Improving the Quality of Healthcare Through Better Information: MediSource Progress Report*, Denver: Multum, technical progress report submitted to Geo. Wash. Univ.

Shea, S., et al. 1996. A Meta-Analysis of 16 Randomized Controlled Trials to Evaluate Computer-Based Clinical Reminder Systems for Preventive Care in the Ambulatory Setting. *JAMIA*, 3(6):399-409.

Silverman, B., et al.1996. *Health Object Library Online (HOLON).* URL=http://www.omg.org/CORBAMED/96-05-14; Gaithersburg: Koop Foundation Inc..

Silverman, B., et al. 1997. Adaptive Planning in Personal Assistant Agents: The Case of Reminders of ToDos-II (R2Do2).URL = http://iai.seas.gwu.edu/HOLON/R2Do2.html; Washington DC: GWU/IAI Tech Report.

Warner, H.R., Olmstead, C.M., Rutherford, B.D. 1972. HELP – A Program for Medical Decision-Making. *Computers in Biomedical Research* 5:65-74.