

From Legacy Systems via Client/Server to Web Browser Technology in Hospital Informatics in Finland

Mikko Korpela

Computing Centre, University of Kuopio, Finland

Abstract

The majority of hospital information system installations in Finland are based on a legacy technology from the U.S. Department of Veterans Affairs (VA). This paper presents an architecture and a tool set which provide a migration path from terminal-based to client/server technology, conserving much of the investments in existing applications. It is argued, though, that a new technological revolution is required in the form of extending the web browser/server technology to operational information systems in hospitals. A blueprint is presented for a further migration path from client/server to browser/server technology. The browser technology is regarded as a major challenge to hospital information systems in the next few years.

Keywords:

Hospital Information Systems, Computer Systems Development, Software Design

Introduction

The majority of the hospital information system installations in Finland make use of the *Musti* family of patient information systems. These systems originate from the mid 1980s and are based on the FileMan database management system (DBMS) and the Kernel utility software developed by the U.S. Department of Veterans Affairs (VA) [1]. The core applications were developed in a joint project by three of the five university hospitals, the Computing Centre of the University of Kuopio, and three vendors. Today half a dozen software firms offer applications packages and systems support within the *Musti* framework.

Three aspects have made the VA technology globally attractive in hospitals. Firstly, FileMan, Kernel and a number of VA's applications packages are available in the *public domain*, i.e. free of charge (see www.hardhats.org). Secondly, they are based on the M technology which makes them highly *portable* from one hardware platform to the other, and *scaleable* from stand-alone Personal

Computers (PCs) to clustered mainframes. Thirdly, FileMan outperforms commercial relational databases since it is a network DBMS, not relational by physical design, and makes use of the efficient physical storage of the M technology. In addition to the 170 hospitals of the VA itself, the same or similar technology is in use in the hospitals of the U.S. Department of Defence and in Finland, Germany, Egypt, Pakistan and Nigeria, among others [2] [3] [4] [5] [6].

However, the VA technology is based on the hardware architecture of the early 1980s, i.e. on-line transaction processing with minicomputers and terminals. This has made it outdated in two respects. Firstly, the processing power of today's PC workstations cannot be utilised. Secondly, modern Graphic User Interfaces (GUIs) cannot be utilised. As a consequence, the further development of the *Musti* applications in Finland stagnated by the turn of the 1990s, in waiting for new solutions.

The leading vendor of laboratory information systems in Finland, Mylab Corporation, together with three university hospitals established a project in 1995 to develop a migration path from the legacy technology to the 21st century. The implementation of the study was assigned to the Computing Centre of University of Kuopio. This paper presents the results of the first phase of the study and a plan for a new phase.

The structure of the paper is as follows. First, the objectives of the study are stated and the methods discussed. The following section presents the results of the first phase, which provides a migration path from the legacy technology to the client/server technology. In the discussion section it is argued that a further migration to the web server technology is required soon. A blueprint of the task ahead is provided. In conclusion, the wider applicability and relevance of the results are discussed.

Materials and Methods

The objectives of the study can be summarised as follows.

- First, the *architecture* for a new generation of

applications was to be developed. After that, a set of systems development *tools and techniques* was to be selected or developed.

- The *client/server technology*, windowing, graphic user interfaces and high-productivity programming tools were to be applied.
- The investments made in the legacy systems were to be *conserved* as much as reasonable. The migration path was to be stepwise so that lengthy delays will be avoided.
- The main focus was to be on the modernisation of *departmental* systems like the laboratory information system [7], since another project had been established already in 1991 to replace the core applications, using a different technology.

In Information Systems research, a wide range of methodological approaches from the contributing sciences can be applied, depending on the case [8]. For our study, the quantitative methods of natural and biomedical sciences, like controlled experiments, as well as the interpretative methods of organisational and social sciences are clearly inappropriate. Since the present study is *constructive research* by nature, a combination of review, analysis and field experiment methods was used.

First, existing and emerging technologies were surveyed which might be relevant in "modernising" the *Musti* applications. The findings were analysed and grouped into three alternative strategies. The board of the project then selected one of the strategies as the basis for further work. The actual systems development tool set was then designed and implemented in a prototyping manner. An early version of the tool set was used in a small pilot project ("modernising" a quality control system for clinical laboratories) by Mylab Corporation, the experience was analysed and a heavily re-designed version was developed.

Results: From legacy systems to client/server

The strategies for modernising the M/FileMan-based legacy systems were analysed in terms of how much of the investment in the existing technology is conserved, as opposed to introducing completely new technologies [9]. The strategies were divided into three main alternatives (Figure 1).

Presentation	M	non-M	non-M
Application Logic	M	M / non-M	non-M
Data Management	M	M	M / non-M
	1	3	2
	All-M	Compromise	Standards

Figure 1: Three strategic alternatives for modernising M/FileMan-based information systems.

In the "all-M strategy", all the main layers of the client/server technology — data management, application logic, presentation — are implemented using systems software which is based in M technology. The idea is to conserve as much of M expertise and applications software as possible. Some administrative information systems in the University of Kuopio had been successfully developed already in this way. The specific GUI software which was considered had the extra advantage of supporting "dumb" terminals also.

The other extreme is the "standards strategy", which tries to make use of industry-standard software tools wherever possible. The idea in this strategy is to benefit from the huge development resources allocated in these tools by the leading-edge software companies. The physical data storage can still be implemented in the efficient M technology. The presentation and application layers are implemented e.g. in Microsoft's Visual Basic in the "fat client" model, and connected to the database by using SQL and ODBC, bypassing FileMan.

In the "compromise strategy", the presentation layer is implemented using industry-standard GUI tools and the data management layer using existing FileMan DBMS. The application logic will need to be partly re-written as event-driven GUI scripts, while some of the existing M code can be retained. The idea behind this strategy is to conserve the best parts of the existing technology and apply industry standard software tools where they are at their best.

The all-M strategy was considered too dependent on a single vendor, although the possibility of using "dumb" terminals would have made the transition much smoother. The standards strategy, on the other hand, was considered too inefficient to large hospitals, and it would have required an almost complete re-programming of all existing applications. The board of the project thus selected the compromise strategy for further development.

The VA had meanwhile also studied the same issues, and also arrived at one form of the "compromise strategy" which was announced in early 1995. More precisely, VA selected Borland's Delphi as the tool for the presentation layer, and decided to develop dedicated messaging software to link the client and the server in such a way that existing M software could be called from the client PC. The latter was tagged the Remote Procedure Call (RPC) Broker [10]. The VA also announced that they would make available readily made Delphi components (FMComponents) to call FileMan functions from the client software.

The Finnish project fully approved of the VA's new architecture and decided to build on it, although the actual software was not released until in December 1996.

Meanwhile we developed a more detailed software architecture (Figure 2) [11]. The main principle of the architecture is to specify as high-level components as possible, from which applications can be composed.

Figure 2 presents the various software layers and elements on the PC client and the M server. There are four layers, separated from each other by well-defined interfaces depicted by dotted lines in the diagram. Starting from the bottom, the *first layer* contains of the FileMan database and the M software which can access the database directly. The external interface of this layer is an Applications Program Interface (API) comprising all the M procedures which have been defined in the Remote Procedure (RP) file. Besides the FileMan and Kernel functions, any existing M application program can be made part of the API by announcing it in the RP file. The access rights of various user groups to execute various procedures are also specified in this file.

The *second layer* consists of the RPC Broker Server on the server computer and the RPC Broker Client on the PC, connected with each other by way of the TCP/IP communications protocol. The purpose of this layer is to "push" the M Procedure Call API to the client side, i.e. to enable the Delphi Pascal programs on the client PC to call the M programs that reside on the server computer. When a client program issues a remote procedure call, the RPC Broker Client sends the call and its parameters as a message to the RPC Broker Server; the latter checks the access rights, calls the M procedure and sends the results back.

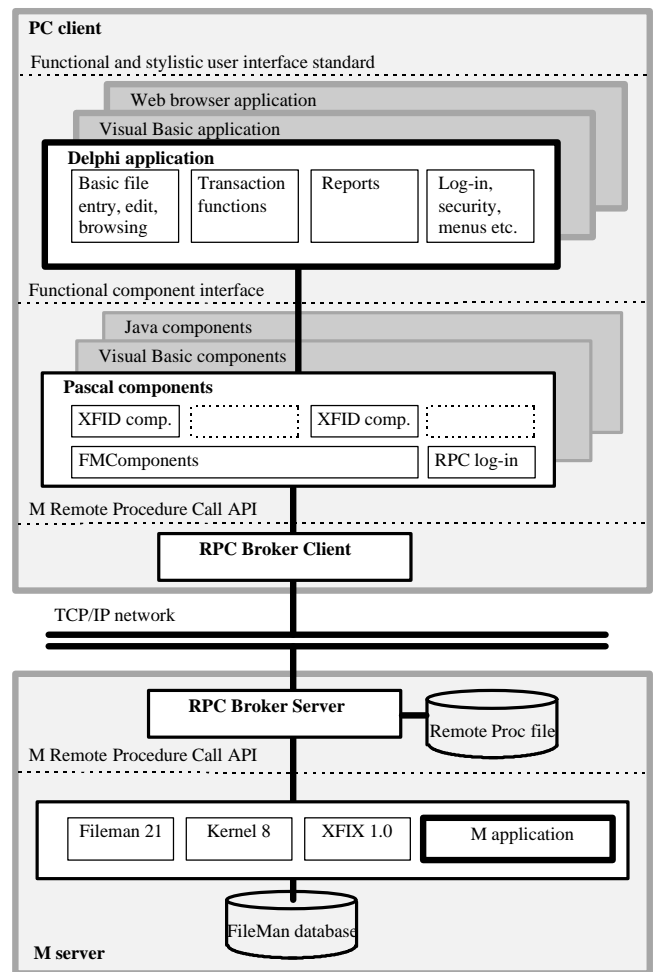


Figure 2: The software architecture.

The *third layer* contains the ready-made components which are available to the applications developers. The components will be discussed in more detail later. Finally, the *topmost layer* is made up of the actual application-specific software. It is only this layer that is directly visible to the user; thus it defines the functionality and style of the user interface.

The application developer's task can be made easier by extracting as much logic as possible from the application layer to the components layer. For that purpose we must identify functions which are common in most applications. In our analysis (see the topmost layer in Figure 2), a large part of all database applications tend to be simple file entry, edit and browsing. Thus we developed high-level standard building blocks, called *XFID components*, from which these routine parts can be composed with no or very little coding. Similarly, various routine reports should be developed easily, so we added components which enable existing reports coded in M being invoked from the client.

With these two classes of simple functionality, in addition to the system services like user log-in, the more routine-like *data-driven* parts of the application can be developed quickly and the developer can concentrate on the *process-driven* parts which need to be fine-tuned according to the

requirements of the user's work process, i.e. the transaction functions.

For the basic file entry, edit and browsing functions we analysed how all the various data structures found in FileMan databases can be visualised to the user in a standard way. As a result we specified a set of visual elements — *tabbed notebook* forms, *subfile list* forms, *zoom* buttons, etc. — by which any database structure can be represented to the user so that she or he can move around it as far as the access rights will allow. These standard elements automatically create a standard "touch and feel" to the user interface of all applications developed with them.

The first version of the tool set made use of the VA's *FMComponents* directly. However, systems developers complained that too much routine programming was needed on the application layer. We then completely re-organised the *XFID components* layer to work more like Borland's standard *database engine* model, used with relational databases. The new model decreased the need for application-level coding dramatically, in comparison to the VA's original way.

Discussion: From client/server to browser/server

The tool set described above was named *FixIT* and officially released in May, 1997. All the *Musti* vendors have obtained the tool set and the first applications are being "modernised" or developed. The first results indicate that it is indeed quite fast to develop visual interfaces to FileMan databases with it, but, on the other hand, there is also a lot of space for improvement and expansion. No real experience exists as of yet about the throughput of the technology in run time, nor about acceptance by hospital users beyond those involved in the pilot project.

It seems, though, that the objectives set in the beginning were reached; an architecture and a set of tools have been created which enable rapid development of modern GUI clients while conserving a great part of the investments in the FileMan databases. There is thus now a migration path from the legacy technology to the client/server technology.

However, the client/server technology is already being challenged by a new "revolution". The avalanche of the World Wide Web (WWW) technology to almost everybody's tabletop has created a situation whereby the *web browsers* are becoming a *de facto* universal user interface technology. In this technology, all that the client computer needs to have permanently installed is the browser, which will then load plug-in pieces of software (*applets*) from respective servers as need arises, in principle from anywhere around the world using the Internet.

So far the web technology has mainly been used just as a big bulletin board. However, several systems development tools companies are working hard on implementing visual programming environments which would make it possible to develop *operational transactions processing applications* as well, e.g. hospital information systems, with this technology. For instance, Borland will release *JBuilder* as a "Java version of Delphi".

The replacement of current "fat" clients by "thin" browser clients in operational information systems has two far-reaching implications. Firstly, the main disadvantage of the client/server technology, i.e. the huge need for system support, will at least be alleviated since the software will be stored in one place only. Secondly, when all applications needed in a given workplace are available as "applets", cheap *Network Computers* (NCs) or *NetPCs* can be used instead of full-blown PCs.

The latter aspect is particularly important to hospitals. University hospitals in Finland currently still have 1,000 to 2,000 "dumb" terminals each, compared to 300 to 1,000 PC workstations each, respectively. The ratio of terminals over PCs is assumably at least equally high in smaller hospitals. All terminals must be replaced by PCs before traditional client/server applications can be utilised throughout, which is a big financial burden with little apparent benefit to the patients. With browser-based applications, the terminals can be replaced by NCs or NetPCs, which cuts the cost significantly.

We envisaged the budding technological revolution in designing our software architecture. With clear functional interfaces between different layers and with as many ready-made high level building blocks as possible, it should be possible to develop a tool set of Java components which is functionally equivalent to the existing Delphi Pascal tool set (Figure 2). Data-driven applications which are now composed of *XFID components* (Delphi-FixIT) with little application-specific code, can be easily re-composed of functionally equivalent *XFIJ components* (Java-FixIT) when the latter will be available.

This approach has a number of major benefits. Firstly, systems developers don't need to sit back and wait for the browser technology to become mature, they can start developing "traditional" client/server applications and later convert the applications to the browser/server technology with relative ease. Thus the investments in learning the GUI technology and object programming, as well as in designing the applications, will not be wasted.

Secondly, the application software is divided into two parts, one residing in the client PC and the other in the M server (thick black boxes in Figure 2). All the "business logic" implemented as M Remote Procedures will be equally useable from "fat" and "thin" clients. Finally, if the Delphi and Java building blocks have the same visible functionality and style, the end users will observe little change from a traditional client/server application to the

browser/server version. Thus even the investments in user training, which are significant indeed when moving from terminals to GUIs, will be conserved.

There is still much technical research work and architectural planning to be done before the browser-based tool set can be implemented in practice. For instance, it is not self-evident where the RPC Broker Client would reside in this architecture. We assess that the detailed architecture can be fixed and a first prototype of the tool set developed by mid 1998, but the first major applications would not be released until about two years later. Of course the timetable depends heavily on the amount of resources allocated in the research and development work. It would also be important to make the collaboration and coordination between the VA and the Finnish development team closer.

Besides the browser technology and the improvements needed in the existing tool set, there is another issue which requires major research and development. In the original *Musti* technology, all applications were integrated by making use of a shared database. In the future, applications must be seen more like autonomous objects (Figure 3). That is, each application contains of some information (database) and methods (program logic) which can be applied to that information; the object communicates with other objects through messaging interfaces.

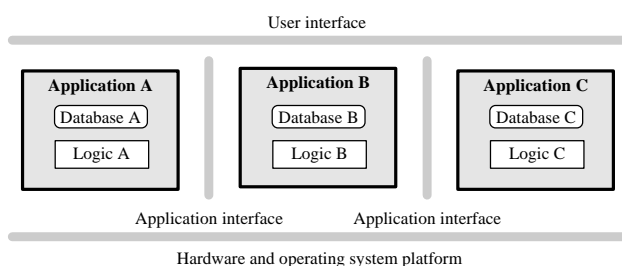


Figure 3: Applications as objects.

Although we recognise that the integrated architecture has some major advantages over the interfaced one [12], in the future hospital information systems will comprise of a number of autonomous applications implemented by different vendors using different technologies originating from different ages.

In Finland there are several projects going on in developing applications interfaces using the Health Level 7 (HL7) protocol. We wish to build on the results of these projects and just develop tools for integrating the interfaces to the internal logic of the application.

Conclusion

The first phase of the project resulted in an architecture and a tool set which provide a migration path from

M/FileMan-based legacy systems to the client/server technology in Finnish hospitals. The applicability of the approach has been demonstrated in small pilot projects; it needs to be validated in more applications projects and in real-life use in hospitals. After that, it should be applied to "modernising" the full range of departmental information systems or to developing new ones.

The first phase also resulted in a blueprint for a functionally equivalent tool set based on the browser/server technology. The proposed solution will offer a smooth further migration path from the traditional client/server technology to the browser-based one. The detailed design and implementation of the tool set will require another project and a couple of years of time.

The modernisation approach presented here should be equally timely and relevant in all other countries where hospital information systems based on the VA technology are being used. However, its dissemination beyond Finland would require close international cooperation and more resources.

The WWW browser technology has so far been used merely as a big bulletin board. The application of the browser technology to the operational information systems in hospitals will be a major challenge of the next few years.

Acknowledgments

The research and development work in 1995–96 was funded by the University Hospitals of Helsinki, Kuopio and Turku as well as by Mylab Corporation. The actual work was conducted by Ms. Kirsi Karvinen, Mr. Mauri Kaatrasalo and Ms. Hellevi Ruonamaa. Ms. Anne Uski of Mylab Corporation provided much valuable critique and advice.

References

- [1] Koskimies J. Use of U.S. Veterans Administration software in Finnish health care. In: Roger FH, Grönroos P, Tervo-Pellikka R, and O'Moore R, eds. *Medical Informatics Europe 85*, Proceedings. Berlin: Springer-Verlag, 1985, pp. 246–250.
- [2] Kolodner RM. *Computerizing Large Integrated Health Networks: The VA Experience*. Berlin: Springer-Verlag, 1997.
- [3] Dolezol W. System protection techniques within the hospital information system at the hospitals of the University of Würzburg. *MUG Quarterly* 1991; 21 (4), 27–32.
- [4] El Hattab O, and Dayhoff RE. Automated medical records technology at the National Cancer Institute-Egypt: A case study in technology transfer. In: Greenes RA, Peterson HE, and Protti DJ, eds. *MEDINFO'95*, Proceedings. Edmonton: HC&CC, 1995, pp. 305–309.

