

# Rejuvenation of Legacy Systems: The Case of M/Kernel Based Hospital Information Systems in Finland

Kirsi Karvinen, Mikko Korpela, Hellevi Ruonamaa  
University of Kuopio, Computing Centre, PL 1627, FIN-70211 Kuopio, Finland

## **Abstract**

M/Kernel based legacy systems in Finnish hospitals are a great asset, but their architecture and user interfaces are outdated. In this paper we present three strategies for rejuvenating the systems, and recommend two of them, depending on the transitional objectives. The study shows that rejuvenation is a feasible alternative for the development of brand new applications. As a conclusion, we suggest concerted international efforts for providing vendor independent tools for new architectures.

## **Introduction**

M technology was introduced in Finland at the turn of the 1980s, when the University of Kuopio was seeking computerized information systems for Health Centres (the primary-care level of the public health care delivery system in Finland). The only feasible system found then was COSTAR, which was translated into Finnish and adapted to local requirements by the university. The system was renamed FINSTAR and handed over to commercial vendors for further installations and development (Jokinen & Hosia, 1987). FINSTAR gradually gained a market share of over 50 % in the Health Centres.

Hospitals also wanted a similar system, and in 1984, a joint project was established between three of the five University Hospitals in the country, two vendors, and the University of Kuopio. The project leaders had heard informally about a database management system called FileMan being developed at the U.S. Veterans Administration. FileMan was obtained to Finland via personal contacts, since it was not yet officially in use at the VA. The project participants translated FileMan, and soon afterwards the entire Kernel, into Finnish. FileMan/Kernel became the cornerstone of the new hospital information system. Because the VA's DHCP core applications were not yet available in the public domain, the admission-discharge, out-patient clinic, laboratory and other applications were developed from scratch or using existing mainframe systems as a starting point (Koskimies, 1985; Soini, 1985).

The resulting family of products was named MUSTI (acronym for MUMPS-based hospital information systems in Finnish). It gained a market share of over 50% of the installed hospital information systems base in Finland. Today FINSTAR and MUSTI are still the most comprehensive packages in their respective fields, and huge amounts of historical patient data are stored in their

databases. However, their architecture is based on a central mainframe/minicomputer and dumb terminals, the user interfaces are archaic, and the systems are not easily integrated with the outside world. Hospitals, health centres and vendors are now in a dilemma: whether to try to "modernize" the existing systems, or to invest in starting from scratch and hopefully converting the historical data over to a new modernized system.

The Computing Centre of the University of Kuopio had meanwhile set up a national FileMan/Kernel support centre in Finland, translating and localizing new versions and offering Kernel support services to M users in Finland. The Computing Centre also provided "International English" versions for the joint hospital informatics project in Nigeria which the university participated in (Daini & al., 1992). In its in-house developed systems for the university administration, the Computing Centre introduced the client/server architecture, using DataTree M and FileMan/Kernel on PC clients and servers, and the SAIC's Hyper-M for graphic user interfaces. Currently these are the only M based client/server and GUI systems in Finland.

Mylab Corporation, the vendor of the Multilab laboratory system in the MUSTI family, together with three university hospitals asked the University of Kuopio to study how the FileMan/Kernel based hospital systems could be "modernized" in a stepwise manner. The study was focused on how to implement client/server architecture and graphic user interfaces while retaining the investments in the FileMan data base. The main results of the study (Karvinen, 1995) are presented in this paper.

We first present the main alternative strategies in general, and describe a few technical solutions for each strategy in some more detail. The strategies and techniques are then compared with each other, and their scope of applicability discussed.

## **Three alternative strategies**

In our study we identified three main strategies, depending on how much of the existing investment in hardware, software and expertise was to be retained, and how comprehensively the new technologies were to be introduced. Figure 1 below illustrates the strategic alternatives.

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

Figure 1: Three strategic alternatives for modernizing M-based information systems.

**Strategy 1: Keep as much M software and terminal hardware as you can.** In this **all-M strategy**, the database, application logic, and presentation logic are all implemented in M. The objective is to move from a centralized architecture to a client/server architecture and a graphical user interface with as few changes as possible to the existing applications. Another important objective is to retain the support of dumb terminals which currently represent some two thirds of the end-user hardware in the hospitals.

**Strategy 2: Use as much standard software tools as you can.** In this **standard tools strategy**, only the physical data storage is in M, while the entire application logic is re-written using an industry standard GUI tool. The clients are linked with the data base server through standard high-level messaging techniques like SQL and ODBC. The objective is to achieve maximum connectivity (openness) and fully benefit from the advantages of commercially available technologies, without exporting the FileMan data bases to a standard DBMS. In this strategy, terminals would not work with the "modernized" applications and would need to be replaced with desktop computers.

**Strategy 3: Keep some M applications software, use some standard GUI tools.** In this **compromise strategy**, the presentation logic is implemented with standard GUI tools, and the application logic is implemented with event driven GUI scripts and M code. The objective is to achieve the main benefits of the two other strategies while avoiding their main drawbacks.

In all three strategies, Windows PCs are used as GUI work stations, and the centralized M database is retained. Of course the fourth alternative would be to replace the M globals with some other technology, but then we would be talking about developing new systems instead of refurbishing the legacy systems. We also regarded that the efficiency of M data bases could hardly be matched by any other technology available — a very important argument in hospitals where the amounts of information and the response time requirements are high.

Possible technological solutions for each of the strategies are presented in the following. The software architectures of the client work stations are illustrated using typical, currently available development tools.

## Solutions for the All-M Strategy

For a closer case study of the All-M Strategy, we selected Hyper-M as the technology. It is originally developed by SAIC (Science Applications International Corporation). When SAIC discontinued Hyper-M development, CDS (Computer Design Systems Ltd.) from the UK, acquired European rights to the software and has since developed it further. It is a tool written in M for developing graphic user interfaces, using M as the scripting language.

Hyper-M has an extraordinary feature of supporting three interface technologies: graphic mode through Microsoft Windows, text mode under MS-DOS, and dumb terminals (DEC VT220 compatible). The drivers for the latter two are built in, while the first one uses the DT-Windows driver by InterSystems. CDS is currently developing a version which supports the InterSystems Visual M technology and the M-WAPI interface (on Unix platforms for instance). All modes share the fundamental Windows "touch and feel" aspects like buttons, pop-up windows, pull-down menus, check and radio boxes and hot keys. Displays designed for terminals run as such on the better interfaces, while the higher end modes provide for extra facilities like mouse operations and bit map graphics.

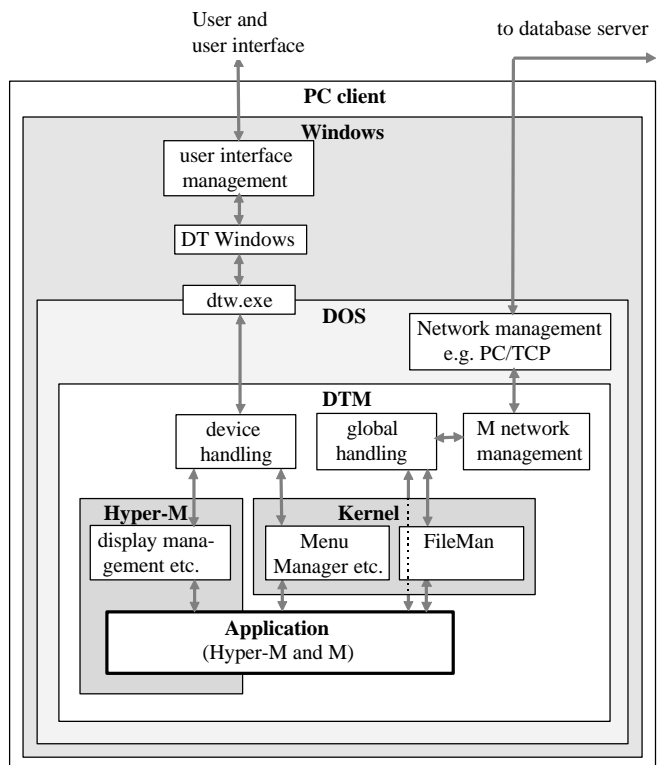


Figure 2: The software layers and communication links needed when a Fileman/Kernel based application is refurbished using Hyper-M.

When an existing FileMan/Kernel based application is refurbished using Hyper-M as a front-end, all the software

runs on the PC client. The client accesses the physical data base through remote SET and KILL commands across a local area network (Figure 2 illustrates the software layers, components and communications links needed).

For minimum changes to the existing applications, the user will log in and select a menu option using the old rolling mode Kernel functionalities. When a new style of an option is selected, Kernel's Menu Manager invokes Hyper-M to manage the dialogue. Newly developed, event-driven scripts will call existing application modules or FileMan directly for database storage and retrieval. GUI facilities can be introduced on an option-by-option basis, rather than across the board at the same time.

When the user has a dumb terminal, the only difference in the software architecture is that Hyper-M will manage the entire user interface, and the MS Windows layer is not needed. Currently Hyper-M runs only on DataTree M on PCs, and thus a separate Hyper-M server PC is needed as a front-end between the terminals (terminal servers) and the VAX VMS or Unix database servers used by hospitals in Finland. Properly designed Kernel-based applications software will run on any of these platforms without changes.

An alternative technology would be Kernel GUI (Ivey, 1995), which we awaited with much enthusiasm in Finland. Like Hyper-M, it is an all-M GUI development tool which incorporates a driver for some "Windows look-alike" functionalities for dumb terminals, as well as an interface to commercially available windowing products (through M-WAPI). It has an extra advantage of being developed by VA and thus being well integrated into FileMan and Kernel. However, VA did not release it for the public domain, and will apparently not support it even internally in the future.

### Solutions for the Standard Tools Strategy

There are many possible industry standard tools for rebuilding the existing M applications. We selected the Borland Delphi with SQL Links as the case for this study. Delphi is a comprehensive GUI applications development tool which has rapidly gained popularity. It uses Object Pascal as the scripting language, and popular press has regarded it as unmatched by other similar tools in terms of throughput.

Figure 3 shows the software layers and components of the PC client in the Delphi-based solution. The application logic, including the functionality previously implemented by FileMan and Kernel, need to be totally re-developed using Delphi's software tools and Object Pascal; correspondingly, there is no M component in the client configuration.

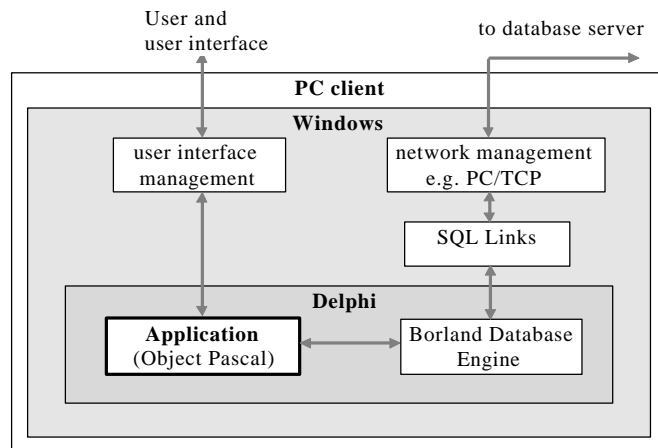


Figure 3: The software layers and communication links needed when a Fileman/Kernel based application is refurbished using Delphi and SQL.

In order to make full use of standard technologies, the connection between the applications logic and the database is implemented by using the SQL messaging technique. To that end, the FileMan data dictionary on an M/SQL database server has to be transformed into a relational data dictionary. There are automated utilities for performing the transformation, but some manual work cannot be avoided. During run time, neither FileMan nor Kernel is needed any more; the M environment on the database server is used only as an SQL engine to access the physical global structures.

### Solutions for the Compromise Strategy

In the compromise strategy, standard tools are used for the presentation logic, while at least parts of the previously existing M software are retained. For the present study, the tool we selected was InterSystems Visual M, which uses Microsoft Visual Basic as the standard GUI development tool. With this combination it is possible to code most of the application logic in either M or Visual Basic.

Figure 4 illustrates the software layers and components of the PC client. The application is divided into two parts, namely the Visual Basic and the M components.

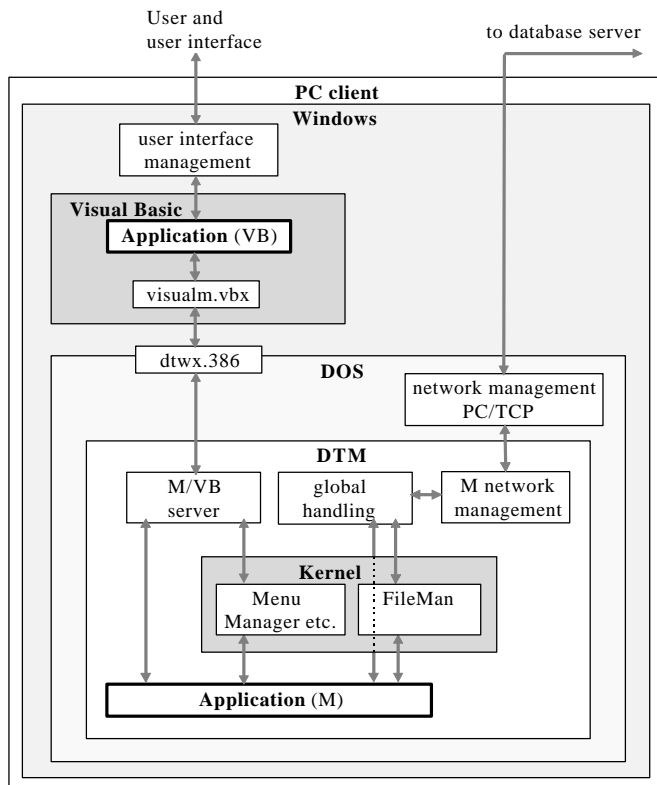


Figure 4: The software layers and communication links needed when a Fileman/Kernel based application is refurbished using Visual M.

The software architecture looks very much like the one of the Hyper-M solution, except that the entire presentation logic resides outside of M. The Visual Basic scripts call M data base or processing modules through a linkage comprising of a custom control on the VB side and a server process on the M side. Only a fixed amount of non-subscripted data can be carried over by each call. The M routines on the client, particularly FileMan, then access the physical data base through remote SET and KILL commands, as in the all-M strategy.

InterSystems has recently released a version which allows the M portion to be moved from the client to the database server. The structure and components of the system remain as in Figure 4.

As in the standard tools strategy, the log-in, menu management, etc. facilities of the Kernel cannot be used any more, because the latter cannot READ and WRITE on the Visual Basic interface. However, in the compromise strategy similar functionalities can be developed with the standard GUI tool, to make use of existing Kernel routines and data structures. To this end, a non-interactive applications programming interface (API) to Kernel and FileMan is required. It became largely available only with versions 8 and 21, respectively, in 1995.

## Comparison of the strategies

The All-M Strategy has two major advantages: it is the only way of enabling the dumb terminal users to share some of the benefits of the graphic user interfaces, and it allows for a gradual refurbishment path through an option-by-option transition to the new interface. It is a democratic strategy because it does not exclude the less facilitated users from new developments. In the same time it does not require vast one-time investments in either hardware replacements or software development.

The main drawback of the All-M Strategy is the instability of the tools available. Hyper-M has proved practicable in the University of Kuopio, but some aspects of it should be further developed. Without a commercial support in the USA, it is too risky an investment for Finnish hospitals as a cornerstone of rejuvenating the VA Kernel based systems. Kernel GUI had trustworthy global development prospects, although it might originally have been technically less advanced than Hyper-M — but, alas, its development was discontinued as well.

The main advantage of the Standard Tools Strategy is that different data bases — M and non-M — can be accessed transparently by the same application, and end users can retrieve *ad hoc* reports from the FileMan database using commercially available relational query tools.

However, the Standard Tools Strategy is not feasible for rejuvenating operational core applications because too much needs to be re-developed and the throughput of the resulting new system might still not be sufficient for critical applications. Instead, this strategy can be considered for *ad hoc* reporting extensions to existing systems.

The Compromise Strategy is intended to provide the best sides of both extreme strategies; an attractive user interface with a reasonable investment of work. If not applied with care, however, it can just combine the worst drawbacks of the extreme strategies; the majority of users (with dumb terminals) are excluded from new applications, but the development of fully modern applications for the PC users may still take much time and result in sub-standard interfaces. As a consequence, both PC and terminal users may be left unsupported for a lengthy period of time.

Particularly, it should be understood that there is actually no truly standard GUI development technology, but a few competing proprietary ones. If you build your applications on Visual Basic but Delphi wins the game, or vice versa, or if indeed the standard GUI development technology in 1998 is Java, then you again have a legacy of old fashioned, non-standard, vendor dependent systems.

The risks of the Compromise Strategy can be decreased if the vendor dependent components are isolated by functional programming interfaces. That is, instead of directly setting and killing globals from a Visual Basic or Delphi script, one should try to call as high level M modules as possible. Then

it might be possible to change the presentation logic when needed, with a reasonable amount of work.

The VA is developing a technology called Kernel Broker, to provide a rather high level of functional modularity, somewhere between the Standard Tools Strategy and the Compromise Strategy. In terms of network architecture, the Broker can be seen as a replacement of SQL as the messaging technique between Delphi and M in Figure 3. That is, the Object Pascal application software can access the FileMan database (and Kernel services as well, in this case) through the Broker. The latter consists of a Pascal part in the client and an M part in the server, communicating with each other by means of the TCP/IP protocol and a non-standard message grammar.

As a rejuvenation technology, however, the Broker falls rather under the Compromise Strategy class, because it allows for much of the existing M software to be retained. From that point of view it can be seen as a replacement of the M/VB linkage in Figure 4. At the time of our study it was not available for testing, however.

The most attractive alternative would be to have so high a level of functional modules that the remaining presentation logic could be easily enough written *both* in a standard tool like Delphi or Visual Basic for PC clients, *and* in an M tool like Kernel GUI or Hyper-M for terminals. Probably some parts of such functionality should be separately coded for different host GUI technologies.

## Conclusion

The study revealed that it is indeed possible to rejuvenate the M/Kernel based legacy systems in Finnish hospitals by retaining the databases and implementing modern client/server and GUI techniques. However, none of the alternative strategies is without some problems.

If the main emphasis is placed on providing a gradual replacement path for dumb terminals, then the All-M Strategy should be selected. The Compromise Strategy is more suitable if the emphasis is on the full utilization of modern GUI tools on an application by application basis, rather than for all users at the same pace. We do not consider the Standard Tools Strategy feasible for operational use.

The success of the All-M Strategy is entirely dependent on the continued availability of a stable, vendor independent M GUI tool which supports terminals (e.g., Hyper-M or Kernel GUI). The full success of the Compromise Strategy is dependent on the availability of high level interfaces which isolate the vendor dependent parts (e.g., Kernel Broker).

In both cases, concerted international efforts would be highly desirable. We have repeatedly seen that technologies developed in one country are not necessarily easily localized to other languages, character sets, etc. In the case of Hyper-

M, the unfortunate geographic split of further development and support made an otherwise workable technology too doubtful for Finnish FileMan/Kernel users. We are apprehensive that even in the case of the Kernel Broker, too much is depending on the good will of the already overburdened VA development teams. There should be a formal international collaboration scheme which would share the expenses and safeguard a continued support.

In Finland, the pilot project participants decided to carry on along the Compromise Strategy. As the next step, a small sample application is being rejuvenated by using Borland Delphi as the GUI tool and hopefully the VA Kernel Broker as the client/server connection tool.

## References

Ivey J: M Window Programming in the VA Kernel Environment. Tutorial in the 24th MTA Annual Meeting, Chicago, June 5–9, 1995.

Jokinen Y, Hosia P: Experiences with a Comprehensive Computer Based Information System for Primary Care in Finland. In: Reichertz PL & al. (eds.), *Present Status of Computer Support in Ambulatory Care*, Springer-Verlag 1987, p. 27–32.

Karvinen K: *Graafisen käyttöliittymän ja asiakas-palvelin-arkkitehtuurin toteutusvaihtoehdot Suomen M-pohjaisten sairaalatietojärjestelmien modernisoinnissa*. University of Kuopio Occasional Reports C. Natural and Environmental Sciences 1. 1995.

Koskimies J: Use of U.S. Veterans Administration software in Finnish health care. In: Roger FH & al. (eds.), *Medical Informatics Europe 85*, Proceedings, Springer-Verlag 1985, p. 246–250.

Daini OA, Korpela M, Ojo JO, Soriyan HA: The computer in a Nigerian teaching hospital: First-year experiences. In: Lun KC & al. (eds.), *MEDINFO 92*, Proceedings, Elsevier 1992, p. 230–235.

Soini E: MULTILAB – A fourth generation laboratory information system. In: Roger FH & al. (eds.), *op. cit.* 1985, p. 567–571.