AN ARCHITECTURE FOR AN INTEGRATED MEDICAL WORKSTATION; ITS REALIZATION AND EVALUATION

Erik van Mulligen

AN ARCHITECTURE FOR AN INTEGRATED MEDICAL WORKSTATION; ITS REALIZATION AND EVALUATION

EEN ARCHITECTUUR VOOR EEN
GEINTEGREERD MEDISCH WERKSTATION;
DE REALISATIE EN EVALUATIE

Proefschrift

ter verkrijging van de graad van doctor
aan de Erasmus Universiteit Rotterdam
op gezag van de rector magnificus
prof. dr. C.J. Rijnvos
en volgens besluit van het College van Dekanen.

De openbare verdediging zal plaatsvinden op
woensdag 1 september 1993 om 15.45 uur

door

Erik Matthias van Mulligen

geboren te Wildervank

Financial support by Hewlett-Packard for the publication of this thesis is gratefully acknowledged.

Cover design: Gert-Jan Slegt, Tecline, Mijdrecht

ISBN 90-9006284-X

Voor Lydia, Jedidjah, Ruben en Joël

CONTENTS

CHAPTER 1

General Introduction

Introduction

This study describes the development of the HERMES integrated medical workstation for the support of patient care and clinical data analysis. This development proceeded in two steps. First, a prototype integrated workstation was developed for the limited domain of support for clinical data analysis. Second, insight resulting from experience with the design and implementation of the prototype, and from the outcome of its formal user evaluation were used as input to design the new HERMES architecture, also intended to encompass the support of patient care.

HERMES offers a solution for the urgent problem in medical informatics of integrating different applications on different hosts. Our approach combines the client-server paradigm with a graphical user interface to provide user-friendly access to the clinician. Its application domain includes both patient care and clinical data analysis.

In this introductory chapter, we will briefly introduce the idea of providing integrated computer support to the clinician and the recent progress made in computer science that enables this novel approach to workstation integration.

Why Integrated Medical Workstations?

Although many hospitals now have a large installed base of PCs and other computers that are connected in a network, and though many powerful applications are available, the computer support for clinicians themselves is still not satisfactory. For many tasks, clinicians have to rely on the assistance of a special staff that has experience with the applications, the communication through the network and the exchange of data between applications.

The problems that clinicians have with the current computer support for their tasks is primarily caused by the following reasons:

- *Not all tasks are covered by one single application*; frequently, different applications (from different vendors) are involved in the support of a clinical task.
- *Not all applications can be accessed from one station*; for some applications, the clinician has to move to another station, due to the fact that either the network connection does not exist or is too difficult to use by clinicians.
- If, however, all applications can be accessed from one station, *data exchange between*

the applications is the next problem that a clinician has to face; differences in the data file format used by the applications can prohibit a smooth and simple data exchange between the applications.

- *The clinician has to remember the command languages and must know how to use the various applications.* Differences in commands, terminology and output presentation force the clinician to concentrate on the applications rather than on fulfilling a clinical task.

In summary, the hardware, software and network is available in many hospitals, but the support of clinical users requires an approach that hides the differences between the applications and the complexity of network access.

This problem has been recognized by many researchers, and various approaches have been proposed to improve this situation. Our solution to this problem is the development of an integrated medical workstation. In this study, an integrated medical workstation is defined as a workstation that enables clinicians to access computers in the network without having to know all network details, and that automatically arranges the needed data exchange between applications (e.g., between a hospital information system and a statistical application). Operation of the application is handled through a uniform user interface at the workstation that provides the necessary mapping between high-level functions visible to the user and what is required by the application, and between the outcomes of applications from some computers in the network, and the graphical representation on the user's workstation.

Historical Background

In the late Sixties, medical informatics started as a discipline to investigate the possibilities of computer applications in health care. This eventually resulted in information systems, for example, for hospitals, specialists or general practitioners, in applications such as those for interpreting ECGs, to analyze images, in expert systems etc. Several of these applications are now available to clinical users for support of their practice and research. This first generation of support systems offered by medical informatics and the biomedical industry can be characterized as being limited to a specific problem domain and not supporting a clinical domain in its entirety. In the present study, we investigated the possibilities of relatively new technologies for starting, what might be called, a second generation of medical informatics support systems that makes a variety of applications directly accessible by clinicians for their tasks in both patient care and clinical research.

This second generation of support systems combines new developments such as network communication technology [1,2], user-friendly interface technology [3,4,5,6], and computer-based patient records [7,8] on an integrated medical workstation [9,10,11]. In this approach, the medical workstation can be seen as the assistant of the clinician that takes care of interaction with all the various applications dispersed through a network, while in principle human support staff is no longer required for routine tasks.

Developments that influence and contribute to integrated medical workstations are:

- *decentralization*; large central systems are more and more replaced by networks of servers, workstations, and PCs. These provide the user with an interface that is often graphical, window-oriented and that can be operated by pointing with a mouse. These computers also facilitate display of images and signals. Such solutions often have a hierarchical structure.
- *network technology*; the change of network use from conventional data communication, such as E-mail and file transfer, to interprocess communication has opened possibilities for distributing applications into processes running on different computers (distributed or network-wide computing) [12] and synthesizing processes into new applications (integration). This distribution of applications into several processes can also be automatically arranged by operating systems [13,14].
- *open systems technology*; increasingly, computing environments consist of hardware and applications from different vendors. Connecting these different computers and applications requires standardization and a different view on software, the so-called open systems technology.
- *object orientation*; this software construction technique provides mechanisms for programmers to abstract from the data, and implementation of functions in objects. Similarly, object orientation can be used to abstract from the data and implementation of functions in applications.
- *standardization*; in order to link systems from different vendors, it is necessary that a

standard is defined for data exchange [15,16] and for accessing functions. Recently, a reference model for distributed computing was proposed [17].

Difficulties with Integration

All these developments together make it technically possible to consider the development of a truly integrated medical workstation environment. The technical problems that have to be faced are the following:

problems related to the integration of applications

- Most applications are not suited for integration in a medical workstation as they are; they do not provide entrances that can be accessed by other applications, and do not offer automatic data exchange facilities.
- The differences that exist between the applications can be considerable. Parameters derived from the data may vary between applications; transferring these parameters from one application to another assumes techniques to rewrite or derive parameters. Sometimes, however, this is hardly possible.
- Standard, transparent network communication between computers of different vendors or even from one vendor is hardly supported yet.
- Standards for data storage, command languages and user interfaces are still not available.

problems related to an integration architecture:

- An integrated workstation has to be flexible enough to support all different applications on the one hand and, on the other hand, provide easy extensions for new applications.
- Both within and outside medical informatics, there is hardly any experience with integrated workstations and their architectures.
- The data formats used by applications are sometimes not documented and cannot directly, i.e., without interference of the application, be accessed or generated.
- Interpretation of graphical output from an application is a laborious task and often requires large programs.

The aim of the HERMES architecture as described in this study is to alleviate all these problems. We will, however, first discuss the various approaches that are possible to solve the problem of integration described above.

Different approaches to integration

The variety of applications and the fast and frequent changes in knowledge and applications, make healthcare one of the domains that can most benefit from integration in workstations.

During the last five years, various integration projects have been started both in the health care domain and outside. These projects can be distinguished along many different axes, such as mechanisms employed and properties. We will now discuss some of these: *pre-facto versus post-facto* integration, *abstraction mechanisms*, *integration levels*, and *communication mechanisms*.

Post-facto versus pre-facto integration

Again, a distinction can be made between types of the projects in how applications have been adapted so that they can communicate with each other. Two approaches can be followed:

- *pre-facto integration.* The applications are modified so that they can understand the commands generated by other applications [18].
- *post-facto integration.* The applications remain unmodified and a layer around the applications provides translations between the communication instruction and the applications equivalent instruction(s) [19].

Abstraction mechanisms

The abstraction mechanism used for distributed databases is the integrated data schema. Through this schema, data from different databases can be combined [20]. Greenes [21] showed in his Explorer-II system how heterogeneous knowledge sources can be integrated into one supportive system for the clinician using different browsing modes for the various knowledge sources. Wiederhold [22] proposed mediators as a mechanism through which a database can be integrated with an expert system.

When multiple applications have to be integrated, the research is more directed towards the integration architecture. Typical properties of such architectures are flexibility and extensibility. The HELIOS project [11] (in health care) and the MINI Software Factory (software development) [23] use a software bus to which all applications are attached. A special piece of code in the application takes care of interaction with the software bus. Communication paths between a client and a server are predefined in these projects. The Common Object Request Broker Architecture of the Object Management Group [24] uses a broker-concept. All applications send their requests to a central broker, which dynamically determines from the request which application can solve it. Another approach is the broadcasting method. A client broadcasts its request to all known servers in the network. Only the servers interested in the request will process it, the others will throw it away [25,26,27].

Integration levels
Applications can differ on various levels. Integration can be seen as an attempt to reduce the differences between applications. Consequently, integration research projects can be categorized according to what type of difference they try to eliminate.

Solving differences between data formats is the category of integration most often present when linking various applications. Data integration offers applications the possibility to transparently exchange data. The Integrated Academic Information Management System (IAIMS) study [28] is an example of a project that integrates various clinical information systems into one large integrated system. Related to this topic is the effort on developing protocols for standard data exchange. These standards evolve for the contents of information exchange [15,29] and for the syntax of information exchanges [16].

When the aim of the integration is also to eliminate differences in command language and to call procedures in other applications, one speaks of functional integration. Although a de-facto standard Remote Procedure Call [30] is currently available, many research projects chose to have their own functional integration methodology. Examples of this type of integration are the software development environment FIELD [25] and the HELIOS environment [11].

Only a few projects related to integration of applications focus on the semantic differences that exist between applications. This ambiguity in semantics is manifest when data from different databases are combined for a particular variable; in each of the databases the meaning of the variable can be different, and combining the data will result in ambiguous data. The Unified Medical Language System [31] proposes a meta-thesaurus that links different terminologies (codings) as used in different databases nowadays. Increasingly, research is devoted to adding semantics to the data. Van den Heuvel et al. have proposed a semantic data model that can be used both for clinical data entry and for clinical data analysis in an integrated medical workstation [32].

Communication mechanisms

Six different approaches [25] have been followed for communication between applications: re-linking, file/pipe, database, shared memory, client-server and broadcast:

- *re-linking*; the source code of the applications is adapted and they are linked into one large application.
- *file/pipe*; applications communicate by writing and reading from a file or a pipe. Synchronization among the applications is important for having a read following a write.
- *database*; the applications store their data and requests in a database. The database management system takes care of the synchronization between the applications.
- *shared memory*; an application writes in a common memory segment its data and instructions together with the addressee. The application addressed will read the data and instructions and process it, and the results will be returned in a similar way. Semaphores are used for synchronization of the write and read operations.
- *client-server*; a network is used as the communication channel between a client and a server. The client writes its instruction which is routed by network software to the server [33].
- *broadcast*; in this schema, a server declares what events it is interested in and a client broadcasts its request to all servers [25,26].

The file/pipe integration (typical a UNIX feature) and the client-server integration styles are the one most frequently used.

Aim and Contents of this Study

In this study, two main questions are addressed: (1) how to build a flexible and extensible architecture for a medical workstation that integrates existing applications within a network, (2) are clinical users supported in their tasks - in this study for clinical data analysis - by this type of integration, and how can this be assessed objectively and quantitatively.

Little initial experience with integrated workstations made us decide to use the approach of prototyping, since this is more adequate for such a new problem, rather than to follow the traditional structured software development paths of analysis-design-implementation-testing. In addition, for the prototype we decided to limit the application domain to clinical data analysis.

We started with the development of a prototype integrated workstation for clinical data analysis that supports data collection, data retrieval, data analysis, data inspection and graphical presentation of results. By choosing this whole trajectory as the aim of the prototype, a representative and yet manageable task was set to investigate and demonstrate the workstation's architecture and capabilities. The first part of this thesis (Chapters 2, 3 and 4) reports on the prototype. This prototype has as much as possible been based on existing applications, and its design and implementation are presented in Chapter 2. The prototype only supports a limited set of statistical and graphical presentation functions, but from the integration of these applications that supported these functions, insight in the problems of post-facto integration has been obtained. Chapter 3 sketches the applications and databases now accessible from the workstation.

A formal user evaluation of the prototype workstation was done for its support of clinical users for clinical data analysis. The evaluation results are described in Chapter 4. An effort was made to measure as objectively and quantitatively as possible the benefits of an integrated medical workstation to the user. A basic difficulty in evaluating integration is that no "zero situation" exists, for comparison with the integrated environment of the workstation. It appeared that the existing literature is very sparse as to this type of evaluations. From this evaluation and our experience with the prototype, the new architecture for its successor, HERMES, has been developed, as described in the second part of this thesis.

This second part of the thesis starts with Chapter 5. An object-oriented framework is presented, the accessor framework, which is the basis of the HERMES architecture. Connections from HERMES with applications are established through these accessor objects. The role of the accessor objects is to hide the workstation from the implementation and access to the applications. The accessor framework is the backbone of the client-server approach as followed in HERMES. Through the accessor framework, clients residing on the workstation can access services that encapsulate the existing applications. The benefits and the impact of the new architecture for support of clinical tasks are shown in Chapter 6, where the connection with current developments in network-wide computing is made. Finally, Chapter 7 shows the new architecture of HERMES with its clients and services. We discuss the communication interface between the clients and services and the realization of the accessor framework as the basis of our new integrated medical workstation environment.

References

[1] Tanenbaum AS. Computer Networks. Englewood Cliffs: Prentice-Hall. 1981.

[2] Halsall F. Data communications, computer networks and OSI. Wokingham: Addison-Wesley Publishing Company. 1988.

[3] Fafchamps D, Young CY, Tang PC. Modelling work practices: input to the design of a physician's workstation. In: Clayton PD, eds. Proceedings of the fifteenth symposium on computer applications in medical care. New York: McGraw-Hill. 1991:788-92.

[4] Cole WG, Davidson JE. Graphic representation can lead to fast and accurate bayesian reasoning. In: Kingsland III LC, eds. Proceedings of the thirteenth symposium on computer applications in medical care. Washington: IEEE Computer Society Press. 1989:227-31.

[5] Shneidermann B. Software psychology: human factors in computer and information systems. Cambridge: Winthrop Publishers, 1980.

[6] Rubinstein R, Hersh H. The human factor: designing computer systems for people. Digital Press, 1984.

[7] Collen MF, Ball MJ. Technologies for computer-based patient records. In: Lun KC, Degoulet P, Piemme TE, Rienhoff O, eds. Proceedings of the seventh world congress on medical informatics. Amsterdam: North-Holland Publishers. 1992:686-90.

[8] Boon WM, Duisterhout JS, Van Ginneken AM. New functional requirements for electronic medical records. In: Lun KC, Degoulet P, Piemme TE, Rienhoff O, eds. Proceedings of the seventh world congress on medical informatics. Amsterdam: North-Holland Publishers. 1992:686-90.

[9] Young CY, Tang PC, Annevelink J. An open systems architecture for development of a physician's workstation. In: Clayton PD, eds. Proceedings of the fifteenth symposium on computer applications in medical care. New York: McGraw-Hill, 1991:491-495.

[10] Van Mulligen EM, Timmers T, Van den Heuvel F. A framework for uniform access to data, software and knowledge. In: Clayton PD, eds. Proceedings of the fifteenth symposium on computer applications in medical care. New York: McGraw-Hill, 1991:496-500.

[11] Degoulet P, et al. The HELIOS european project on software engineering. In: Timmers T, Blum BI, eds. Software Engineering in Medical Informatics. Amsterdam: Elsevier Science Publishers, 1991:125-37.

[12] Boyanov K, Angelinov R (eds.). Network information processing systems. Proceedings of the IFIP TC6/TC8 open symposium on network information processing systems. Amsterdam: Elsevier Science Publishers. 1989.

[13] Tanenbaum AS, Van Renesse R. Distributed operating systems. ACM Comput. Surv. 1985 17(4):419-70.

[14] Rozier M, Martins JL. The CHORUS distributed operating system: some design issues. In: Distributed Operating Systems. Theory and Practice. Heidelberg: Springer-Verlag, 1987:262-87.

[15] Ostler DV, Harrington JJ, Hannemyr G. A common reference model for healthcare data exchange - P1157 medix system architecture. In: Miller R, eds. Proceedings of the fourteenth annual symposium on computer applications in medical care. Los Alamitos: IEEE Computer Society Press, 1990:235-8.

[16] Information technology - open systems interconnection - specification of abstract syntax notation one (ASN.1), ISO/IEC 8824: 1990(E). ISO/IEC Copyright Office, Switzerland.

[17] Bowen D. Open distributed processing. Computer Networks and ISDN Systems. 1991;23:195-201.

[18] Hsu C, Skevington C. Integration of data and knowledge in manufacturing enterprises: a conceptual framework. Journal of Manufacturing Systems 1986;6:277-85.

[19] Power LR. Post-facto integration technology: new discipline for an old practice. In: Ng PA, Ramamoorthy CV, Seifert LC, Yeh RT, eds. Proceedings of the first international conference on systems integration. 1990:4-13.

[20] Safran C. Using routinely collected data for clinical research. Stat Med 1991;10:559-64.

[21] Greenes RA. "Desktop knowledge": a new focus for medical education and decision support. Meth Inform Med 1989;28(4):332-9.

[22] Barsalou T, Wiederhold G. Knowledge-directed mediation between application objects and base data. In: Proceedings of the Working Conference on Data and Knowledge Base Integration. 1989.

[23] Fournier F. The Mini software factory: development of kernel mechanisms around process modeling and software bus techniques. Journal of Systems Integration, 1992;2:145-67.

[24] Soley RM (ed.) Object Management Architecture Guide. Object Management Group, Document 92.11.1.

[25] Reiss SP. Connecting tools using message passing in the Field environment. IEEE Software, 1992;4;57-66.

[26] Gagan MR. The HP softbench environment: an architecture for a new generation of tools. Hewlett Packard Journal. 1990:41(36).

[27] Tang P, Annevelink J, Fafchamps D, Stanton WM, Young C. Physicians' workstations: integrated information management for clinicians. In: Clayton PD, eds. Proceedings of the fifteenth annual symposium on computer applications in medical care. New York: IEEE Computer Society Press. 1991:569-73.

[28] Lunin LF, Ball MJ. Perspectives on integrated information management systems (IAIMS). Journal of the American Society for Information Science. 1988;39(3):102-12.

[29] Simborg D. An emerging standard for health communications: the HL7 standard. Healthcare computing and communications. 1987.

[30] Robinson D. Remote procedure call: a stepping stone towards ODP. Computer networks and ISDN systems. 1991;23:191-4.

[31] Humphreys BL, Lindberg DAB. Building the unified medical language system. In: Kingsland III LC, eds. Proceedings of the thirteenth annual symposium on computer applications in medical care. New York: IEEE Computer Society Press. 1989:475-80.

[32] Van den Heuvel F, Timmers T, Van Mulligen EM, Hess J. Knowledge-based modelling for the classification and follow-up of patients with congenital heart disease. In: Lun KC, Degoulet P, Piemme TE, Rienhoff O, eds. Proceedings of the seventh world congress on medical informatics. Amsterdam: North-Holland. 1992:478-82.

[33] Sinha A. Client-server computing. Commun. of the ACM, 1992;35(7):77-98.

CHAPTER 2

A Prototype Integrated Medical Workstation Environment

E.M. van Mulligen[1,2], T. Timmers[1], F. van den Heuvel[1,3], J.H. van Bemmel[1]

[1]Dept. of Medical Informatics, Erasmus University Rotterdam, The Netherlands
[2]University Hospital Dijkzigt, Rotterdam, The Netherlands
[3]Sophia Children's Hospital, Rotterdam, The Netherlands

(Computer Methods and Programs in Biomedicine 1993;39:331-341)

Abstract
In this paper the requirements, design, and implementation of a prototype integrated medical workstation environment are outlined. The aim of the workstation is to provide user-friendly, task-oriented support for clinicians, based on existing software and data. The prototype project has been started to investigate the technical possibilities of graphical user-interfaces, network technology, client-server approaches, and software encapsulation. Experience with the prototype encouraged discussion on both the limitations and the essential features for an integrated medical workstation.

keywords: workstation, graphical user-interface, encapsulation

Introduction

Last decade's information technology is now rapidly ingressing health care. Professionals are challenged to employ personal workstations, graphical user-interfaces, network technology, relational database management systems, and a wide spectrum of data processing applications. However, many clinicians cannot take full advantage of this new information technology, since its use demands considerable operational knowledge. Stated differently, the current approach of information technology is still primarily tool-based, while the professional's focus on this technology tends to be task-oriented. Consequently, most clinicians favor the paper record because it is less time-consuming, certainly easier to use, although systems now provide medical data, signals, and images. The data stored in most hospital information systems are still foremost intended for administration, billing, scheduling, and resource allocation instead of health care support.

The Medical Workstation project (called HERMES-MW2000) ultimately aims at providing the clinician a personal workstation from which all data, knowledge, and functions can be accessed in a user-friendly way for research and patient care. This workstation should provide the operational knowledge to access the data for various tasks: patient care, clinical research, and education. Starting with existing applications and databases, the workstation acts like a shell around all already existing and future software systems through which clinicians can uniformly address all these data and functions in a network.

In the following sections, the various levels at which integration can be obtained will be outlined. Related research will be discussed according to the levels of integration involved and the ability to deal with existing applications. This is followed by an outline of the integration architecture implemented in the workstation and the applications initially selected. The Conclusion section will elaborate on the usability and the necessary modifications of this architecture and ends with an evaluation of both the integration methodology and the prototyping approach.

1 Levels of integration

The software process outlined above is often referred to as integration: bringing together different, heterogeneous resources at one location, thereby eliminating the inter-resource differences. The differences can be clustered in different categories of integration. For each category, an abstraction layer can be defined that hides the dissimilarities from the user. For HERMES-MW2000 five categories can be distinguished. For some (parts) of the abstractions, current standards are available and the workstation has adopted these standards as much as possible. The five integration categories constitute a subdivision of the application communication layer of the OSI/ISO seven-layer network-communication model [1] (Figure 1). The following integration categories have been discriminated.

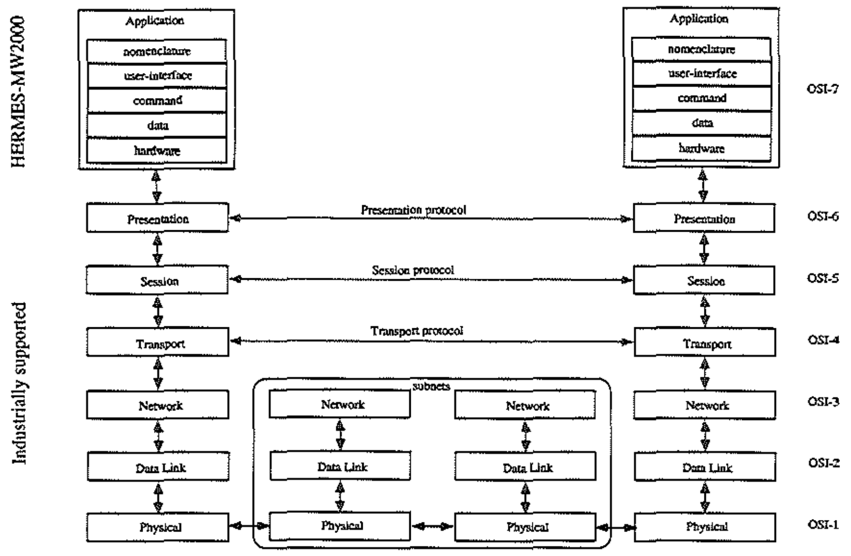


Figure 1. The Integration Model as extension to the OSI Communication Reference Model. The HERMES-MW2000 defines a communication protocol at application level for the various components of an application.

1.1 Hardware integration

This overall integration category conceals the differences between the computer systems in a network. These differences are most profound in the distinction between local and remote processing, and local and remote file access. Basically, one has to utilize a network protocol for starting remote jobs and accessing remote files. Ideally, a commercially available network protocol should eliminate these distinctions, offering one virtual processor (or processor pool) and one virtual file system. Although some commercial network protocols nowadays do offer file system integration, these protocols are limited to only a small range of hardware. Our integration project adopted a standardized network protocol (Arpa/Berkeley) that in itself also does not offer this abstraction [2]. Therefore, an abstraction layer has been defined which, based on the standard network protocol, eliminates these local-remote discrepancies.

1.2 Data integration

The second abstraction category deals with the various coexisting data formats. When using a range of applications to solve a particular task, it is necessary to have facilities for exchanging data between the applications. This exchange of data is hampered by the different data formats of the applications and the lack of common data interchange format that can be used by all applications. Therefore, many clinicians need to re-enter the data in the proper format.

In addition, some data are not available as data files but can only be accessed through a database management system. Exchange of data from a database management system to an application requires the database query to be stored as part of the data format. For HERMES-MW2000, a common data storage format has been defined that contains entries for storing database queries. This data format is used as an intermediate storage format (ISF); exchange of data between applications then always goes in two steps: from the source format to ISF and from the latter to the target format. A software data manager takes care of the translation of data between the ISF and the application formats.

1.3 Software integration

Each application has its typical command syntax and commands. For a clinician interacting with several applications at the same time, these differences are annoying and confusing. Moreover, for a clinical task one would prefer to issue command sequences rather than individual commands. The software integration abstraction defines one command structure with a unique collection of command sequences, which can be expanded to batch programs or series of keystrokes.

1.4 User-interface integration

Each application produces output, either in a file (batch mode) or on screen (interactive mode). The first achievement of this integration is to abstract from the typical terminals required by the various applications. Each application runs on the same display, independently from the typical terminal or graphics adapter. To support this, the workstation provides terminal emulators that simulate these devices. In addition, the user-interface has to handle camouflage of the different output formats and yield a consistent output format. Difficulties arise when integrating interactive applications. Interpretation of generated device control codes and translation to the general output format is a cumbersome and complex task. In our integration project, the latter abstraction has been limited to batch-oriented applications, while each interactive application runs in a window on one display.

1.5 Nomenclature integration

The reduction of inter-application distinctions immediately introduces the danger of semantic misinterpretation. Many terms and headings are interpreted in the (application) context in which they appear and elimination of this visible context may cause errors. The nomenclature abstraction maps the different vocabularies to a general uniform vocabulary, which is preferably standardized. This general vocabulary includes a data dictionary and a medical data model. Such data dictionaries can be user defined.

2 Related research

Although much research has been directed towards integration [3], primarily only a subset of all integration categories was covered. Data integration has gained much attention lately; multidatabase management systems and distributed database management systems combine hardware integration with data integration [4]. The topics covered in this area are more related to schema integration and query optimization than to abstraction mechanisms. Wiederhold [5] and Greenes [6] focus on access of data from within applications and provide common data abstraction mechanisms such as mediators or daxels (data access elements), accomplished by adapting existing applications or developing new software. The Helios project [7] aims at network, data, and software integration and employs a software bus as the communication interchange channel among applications; individual applications have to be adapted to connect to this software bus. The main research focus of IAIMS (Integrated Academic Information Management Systems) is the inter-connection of existing systems [8]. The first phase accomplishes a network, connecting various computers and offering clinicians access to all these systems from each workstation (i.e., the first level of user-interface integration). The next phases aim at data integration, software integration, and nomenclature integration and will be mainly achieved in new software [9,10].

3 Development model

The HERMES-MW2000 project aims at facing all categories of integration, starting with existing applications and avoiding any modification of the applications. Critical to this process of integration is the strategy through which it is achieved. Two opposing approaches can be pursued, i.e., a top-down approach starting with a common integration mechanism and working down to applications and data, and a bottom-up approach, starting with applications and constructing hierarchies of abstraction layers around these applications until one ends up with one top layer. The first approach starts with a sound theoretical mechanism, but the danger is that it lacks a practical reflection. The second method commences with a typical application domain, but there is a danger that the end solution might be constrained to integration of those specific applications only. Both strategies tend to have a breadth-first, or total solution orientation, making corrections awkward and a waste of time. For HERMES-MW2000, a depth-first approach has been utilized by implementing only a limited but representative set of applications according to a general integration mechanism. Deliberate selection of the applications prevents a too specialized and narrow solution, whereas the common integration mechanism can probably be utilized and tested for other applications as well. In the next sections, both the architecture (integration mechanism) and the applications initially selected will be discussed. The Conclusion section will elaborate on the general usability and the necessary modifications of this architecture. Finally, both the integration methodology and a prototyping approach will be discussed.

4 Architecture

Current integration research mainly focuses on new developments that obey the practice of open-systems technology, network dispersion, and modularization. Unfortunately for integration, the medical milieu already has a vast amount of software, data, and knowledge resources available in closed, often isolated systems that can not fulfil new software requirements. The advent of powerful workstations enhances the possibilities to bridge this gap between what is offered and what is desired. The architecture of the HERMES-MW2000 enhances existing applications with the possibility to cooperate with new applications that satisfy criteria of openness, network dispersion, and modularization. In addition, the architecture can easily be extended with new (existing) applications.

Our integration project has been arranged around a client-server mechanism. The client, a graphical user-interface, sends high-level commands to a server. These commands are stored in a database, together with associated parameters, a full (medical) name and a specification of what groups of users are entitled to use it. When activated, the command is transmitted to the server, and the server will properly address the application.

In order to address existing applications that are not immediately able to respond to the high-level commands and cannot utilize HERMES-MW2000's ISF data files, two facilities were added: a data translation facility (DTF) and a command generation facility (CGF). The DTF translates data from the workstation's ISF to the application's format and vice versa, and the CGF expands a macro command to a series of keystrokes or batch-commands for the application. Both facilities employ the network facility (NF) to address files on distant computers and to start applications remotely. Both DTF and CGF encapsulate an existing application, equip it with a standard interchange interface and free the outside world, i.e., the user, from its internal details (Figure 2).

The medical data model can be used by all applications (or encapsulators) to provide the user (and the application) with a standard vocabulary. The workstation's data model has been organized according to a relational structure in which information about each individual attribute can be stored. This information includes a data dictionary, code lists (and their associated meaning), and descriptive names for the attributes. The model has no entries for specifying a computational relation between two attributes, nor has it synonyms. The pretty names can be changed independently by the users, and permit a first level of nomenclature abstraction.

4.1 Applications

The prototype HERMES-MW2000 has mainly been accomplished with already existing applications. A representative range of different applications was selected to test the client-server concept, the DTF and CGF encapsulation, and the ISF definition. Part of the applications may reside on remote computers, linked with the workstation by way of the NF. The idea of a medical data model has been tested in cooperation with existing applications.

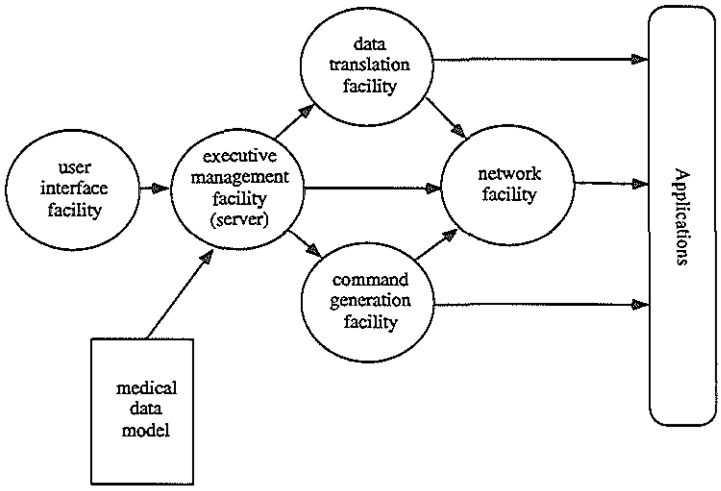

data translation facility
user interface facility
executive management facility (server)
network facility
command generation facility
medical data model
Applications

Figure 2. Overview of the architecture of the prototype HERMES-MW2000. Via the user-interface facility, a user can order a function and activate the executive management facility to solve the request. Subsequently, the executive management facility (EMF) will activate the data-translation facility and the command-generation facility to create a data file in the proper format and a command file with the correct instructions. Via the network facility, this is sent to remote computers and the application is activated. The medical data model is used to translate descriptive names of data attributes to the different attribute names used in the various applications.

4.2 Databases

The workstation has been developed primarily with the aim to support clinical research. The large data sets involved in clinical research and the preferred interactive response, led to the requirement of a rather static clinical research database on the workstation, for which INGRES was selected. During its analysis, a research database should not be changed and no new data should be added (static), in contrast to a clinical database that is continuously updated. Before the analysis, data are most often transferred and mapped from dynamic clinical databases to this static research database. During this data transfer process, the data dictionary part of the medical data model is automatically constructed. The user is prompted to specify the codelists and to identify the keys. All other information is obtained from the data transfer. Currently, data can be imported from a Hospital Information System, from the cardiology departmental information system, from a general practitioner's information system (such as ELIAS [11]) and from widely known systems as dbaseIII, Lotus123 or SPSS. SQL-based databases can also be imported directly into the research DBMS. For the prototype, eight clinical databases were constructed on the HERMES-MW2000 for different research projects.

4.3 Analysis

The statistical package BMDP has for a major part been encapsulated in the HERMES-MW2000 [12], while it is operational on a different computer in the network. BMDP is a batch-oriented application that has an extensive and very specific command language. Its output is written into a file and transformed into graphical output (using OSF/Motif) by HERMES-MW2000. BMDP's descriptive statistical modules and survival-analysis module are currently operational in this way. Additional statistical modules cover the graphical exploitation of cross-tables and confidence intervals with P-values. The former module has been especially developed for the HERMES-MW2000 users and directly accepts high-level commands and ISF data files, rendering encapsulation superfluous. The latter was developed by our department of Epidemiology and has been encapsulated.

4.4 Presentation

The interactive use of the HERMES-MW2000 demands graphical presentation of results. Two approaches have been followed: utilization of an existing application and specific innovative developments. The existing application selected for graphical presentation is Harvard Graphics. In an MS-DOS emulator, this interactive application has been made accessible in the HERMES-MW2000. The CGF expands macros such as for histograms and pie-charts to the keystrokes necessary for Harvard Graphics, even when the application runs under MS-DOS, and the CGF under UNIX. The second approach has been employed for the data inspection module, which includes histograms and tables.

4.5 Images and signals

Besides ASCII data, the prototype also contains facilities for displaying 2-D or 3-D images (CTs, MRIs, Angiograms, nuclear images) and signals (ECGs). The integration of these modules has been limited to the user-interface level: on each workstation these facilities can be called and the output is shown in a window. All facilities have adopted the OSF/Motif user-interface.

5 Implementation

Each of the integration layers is based upon a regular interchange definition for data and commands. These definitions are the common interface for the applications and provide a means for uniformly accessing data, knowledge, and functions.

5.1 Intermediate storage format

The data format has been defined under the name internal storage format (ISF). This format consists of three files, one containing an SQL command, another containing the format itself, the number of variables and the variable names in the data model, and the third file containing the actual data. The command language contains a set of macro commands together with parameters. This set of macro commands can be extended with new commands; its parameter-passing mechanism complies with the UNIX convention.

Each existing application is encapsulated by two drivers: a data translation facility and a command generator. These drivers are activated by the HERMES-MW2000 to transform the data from the ISF format to the application format and vice versa, and to expand a macro command into a series of keystrokes or commands for the application. Activation of a macro command automatically activates the data translation driver and the command generator, which in their turn activate the application. The network layer supports network-wide activation of these drivers and the application. Translated data files are transferred through the network while the command generator is started remotely by the HERMES-MW2000 itself.

Macro commands can be activated graphically on the workstation display screen. In an X11-based OSF/Motif user-interface, the descriptive names of the macro commands are depicted by button widgets that can be activated by pressing a mouse button. The command buttons are organized in a hierarchy of menus. The definitions of macro commands and their hierarchical organization in menus is stored in a database. New macros can be added by editing the database. Activation of a button causes the associated command to be transmitted to the HERMES-MW2000 server. The server then will search its database to obtain the executable shell script.

This server software, called the executive management facility (EMF), accepts the macro commands through a shared memory segment. The user-interface client inserts the command in the shared memory segment and sends an activation signal to the EMF. This EMF then reads macro command and starts an associated UNIX script, which contains commands for obtaining the data file, possibly through the network, and for (remotely) activating the command generator and the application. These UNIX scripts contain the necessary commands for testing the presence of already transformed files.

5.2 Standards

For the development of the workstation, it was decided to adopt current standards as much as possible. The following standards have been adopted in the HERMES-MW2000 environment:

(1) UNIX system V as the workstation's operating system,
(2) Arpa-Berkeley network standards,
(3) X11 OSF/Motif as the graphical user-interface management system,
(4) ANSI-C programming language for new developments,
(5) SQL for expressing data selections to a database management system,
(6) ASCII storage format to facilitate easy transmission to other computers.

Starting with these standards, migration of the HERMES-MW2000 approach to different hardware containing a UNIX environment can be accomplished with a minimal effort [13].

5.3 Tools

The HERMES-MW2000 environment has been developed with a number of standard software tools. These tools are mainly directed at the construction of data-translation drivers and command generators. With the UNIX system V 'awk' tool [14], data translation programs can be developed at a high conceptual level. The 'awk' programs specifies a series of rules that can be activated conditionally. Command generators use a template command script and an associated UNIX system V 'sed' program, which replaces the variable parts of a template script with the actual parameters.

In addition, a graphical editing tool has been developed to compose a data model. Two separate tools are available: a simple tool for modifying the pretty names associated with the attributes and an advanced tool for specifying a complete object-oriented data model [15].

6 Conclusion

The strategy of encapsulating existing applications in a network environment can be followed successfully: current workstations are powerful enough to overcome the overhead caused by the encapsulation layers. Despite the power of present workstations, the success of this approach is limited unless tools are developed that support the developer for creating and modifying the necessary data translation and command generation facilities; since the core of existing applications changes over the years, as necessarily do their application interfaces, the workstation developer should be able to rely on graphical tools to adjust the encapsulation layer adequately.

Preferably, these tools should maintain a database in which all access details of the applications are stored. Updating the encapsulation layers would require a tool to update the contents of that database. Current research in the HERMES-MW2000 project aims at the provision of these tools, making the encapsulation layers more widely and easier available.

In addition to the support for the development of tools, the client-server architecture could be extended to a multi-client/multi-server architecture. Currently, standards are evolving for network-wide client-server communication, such as the TCP/IP-based Arpa-Berkeley sockets. Utilization of this type of client-server architecture could diminish the current problems with hardware integration and offer a virtual one-machine concept. The application-specific information currently stored in scripts and accessible by the central EMF server, could in this architecture be deferred to the application's encapsulation server and dramatically reduce the management complexity.

Current experience with the ISF has learned us that this format should be extended to contain a variety of other data formats as well: binary data files and free-formatted data files should also be covered by the ISF. Preferably, the three format files should be combined into one file, reducing the chance of incorrect ISF data files. Although the HERMES-MW2000 offers users a higher level of support for their tasks than originally provided by the individual applications, future research should be directed at the definition of a user model for the various tasks of a clinician. This research is strongly related to the desk-top metaphor applied, and various different approaches should be investigated [16].

The effort, necessary to establish a user-friendly interface around existing applications, is considerably less than the development of these applications from scratch; evaluated software is precious and its maintenance is even more precious. The performance of current workstations is reasonable to surmount the overhead involved with the extra layers for software integration.

Although the technical integration layers have been established and provide an open, integrated workstation environment, research should be directed towards how to accomplish true clinical task-orientation and to establish which user-interface best fulfills the user's needs.

Extra attention should be paid to solutions that manage changes and provide dynamic extensions. New applications should be incorporated in the HERMES-MW2000 without modifying the already existing applications or encapsulation layers.

Finally, the integration architecture should be extended to have entries for other integrated architectures: bridges that transform messages between the various systems that currently evolve.

References

[1] H. Zimmermann, OSI reference model - the ISO model of architecture for open systems interconnection, IEEE Trans. Commun. 28 (1980) 425-432.

[2] Hewlett Packard Company, Using Arpa services (February 1991).

[3] L.R. Power, Post-facto integration technology: new discipline for an old practice, in: Proceedings of the First International Conference on Systems Integration, Morristown, New Jersey, 1990, pp. 4-13 (IEEE Computer Society Press, Los Alamitos, Ca., 1990).

[4] T.A. Landers and R.L. Rosenberg, An overview of multibase, in: Distributed Data Bases, ed. H.J. Schneider, (North-Holland, Amsterdam, 1982).

[5] T. Barsalou and G. Wiederhold, Knowledge-directed mediation between application objects and base data, in: Proceedings of the Working Conference on Data and Knowledge Base Integration, October, 1989.

[6] R.A. Greenes, Promoting productivity by propagating the practice of "plug-compatible" programming, in: Proceedings of the 14th Symposium on Computer Applications in Medical Care, Washington DC, November, 1990, pp. 22-26 (IEEE Computer Society Press, New York, 1990)

[7] P. Degoulet, F.C.J. Coignard, M.C. Laurent, L. Lucas, M. Ben Said, H.P. Meinzer, U. Engelmann, A. Springub, R. Baud, J-R. Scherrer, The HELIOS European project on software engineering, in: Proceedings of the IMIA Working Conference on Software Engineering in Medical Informatics, Amsterdam, October, 1990, pp. 125-137 (North-Holland, Amsterdam, 1991).

[8] L.F. Lunin, M.J. Ball, Perspectives on Integrated Information Management System (IAIMS). J Amer Soc Inf Science, 39, 3 (1988) 102-112.

[9] S. Johnson, C. Friedman, J.J. Cimino, T. Clark, G. Hripsack, P.D. Clayton, Conceptual data model for a central patient database, in: Proceedings of the fifteenth Annual Symposium on Computer Applications in Medical Care, Washington DC, 1991, pp. 381-385 (McGraw-Hill, New York, 1991).

[10] J.J. Cimino, Representation of clinical laboratory terminology in the unified medical language system, in: Proceedings of the fifteenth Annual Symposium on Computer Applications in Medical Care, Washington DC, 1991, pp. 199-203 (McGraw-Hill, New York, 1991).

[11] J.S. Duisterhout, B. Franken B and F.S.C. Witte, Structure and software tools of AIDA. Comp. Meth. Progr. Biomed. 25 (1987) 259-274.

[12] W.J. Dixon, BMDP statistical software manual (University of California Press, Berkeley, 1981).

[13] E.M. van Mulligen, Migration of the MW2000 from HP9000 to a PC environment, October, 1991 (Internal report).

[14] A.V. Aho, B.W. Kernighan and P.J. Weinberger, The AWK programming language (Addison-Wesley, 1988).

[15] T. Timmers, E.M. van Mulligen and F. van den Heuvel, Integration of an object knowledge base into a medical workstation, in: Proceedings of the fifteenth Annual Symposium on Computer Applications in Medical Care, Washington DC, 1991, pp. 654-658 (McGraw-Hill, New York, 1991).

[16] L.A. ten Horn, Changes in tasks and task requirements of information technology experts and departments in view of end user computing, in: Proceedings of the 20th ICA Conference, Bonn, October, 1986, pp. 133-142.

CHAPTER 3

Applications And Databases In An Integrated Medical Workstation

E.M. van Mulligen[1,2], T.Timmers[1], J.H. van Bemmel[1]

[1]Dept of Medical Informatics, Erasmus University Rotterdam, The Netherlands
[2]University Hospital Dijkzigt, Rotterdam, The Netherlands

The Netherlands Journal of Cardiology, 1993;2:67-72

Abstract

Clinicians are more and more confronted with a wide variety of databases and programs. Often, there are large differences between the commands the databases use for retrieving data and the commands that can be used to start programs. In addition, databases are located on different computers and networks are used to transport data between different computers; access through the network is in general a very complex task for end-users. The HERMES integrated medical workstation aims at hiding from the user all these differences between databases and programs and provides a user-friendly access through the network. Research has been directed towards a flexible and extensible architecture for linking all these databases and programs. Central in the architecture is a medical data model that facilitates multi-database data retrieval and supports data entry. The first prototype of HERMES has been realized for use in Cardiology.

keywords: integration, workstation, clinical research

Integration

Monday night, 8.00 hrs p.m., Department of Cardiology. Assistant X has found some time to continue his research study on survival of hypertrophic myocardiopathy. While at home during the weekend he has updated his database of some 100 patients with the latest findings on mortality from the municipality and findings from the paper medical record. The update must be imported into the Workstation for analysis. He has brought the dBaseIII files with him, logs onto the Workstation and gets access to his own research directories. After typing in the name of the file, the database is copied into the Workstation research database, and the analysis can start. First, X makes a selection from the database of the variables and patients related to follow-up. The variables are presented on the screen by their usual medical name; and conditions are entered by selecting from code lists that are stored in the system, so X does not need to remember them. The selections are made by means of a "mouse", a device that allows one to point at spots on the screen. Subsequently, a contingency table is created of "Syncope during Follow-up" against "Status" (i.e., alive or deceased). By simply "pointing and clicking" with the mouse at one of the appropriate "menus", a program is started that computes and shows the corresponding P-values and rate ratios. Because X is to present his results at the next staff meeting, he uses another Workstation facility to prepare slides of the contingency tables in Harvard Graphics. Again, this is done automatically, without the need for retyping the data.

In the earlier intermediate analyses, the mortality rates were surprisingly low. X is interested to know what a formal survival analysis reveals. X selects the "Life table and Survival analysis" option from the "Analysis" menu, and the Workstation automatically starts up module 1L of the BMDP Statistical Package. Subsequently, the Workstation graphically presents the result on its screen. X decides to consult the Department's statistician Y for further analysis, but right now he makes a printout of the graphs. Maybe Y needs to make a more extensive analysis of the data, using the full functionality of BMDP. At least these initial results can be communicated to Y. X drops a message in Y's electronic mailbox, which Y will read next morning when he logs onto the Workstation through the terminal on his desk.
After "playing" with some other "views" on his data, X decides to stop and go home. It is Monday night, 9:00 hrs p.m. [1].

What is integration?

Nowadays, computers are a common appearance in hospitals, but they are mainly used for administrative purposes. The use of computers for storage and retrieval of patient data, for clinical data analysis, biosignal interpretation, image presentation, etc. is increasing in hospitals. Although researchers in medical informatics have started to investigate problems of how to store clinical data and knowledge in a computer [2,3], a more basic problem first of all has to be dealt with: how to integrate. As the paper medical record consists of data from various sources (different departments, laboratories, catheterization laboratories, ultrasound imaging, ECG recording equipment, CT, etc.), the computer-based medical record can also be seen as consisting of data from different computers and different databases. This bringing together of data from different electronic sources is called integration (see Figure 1).

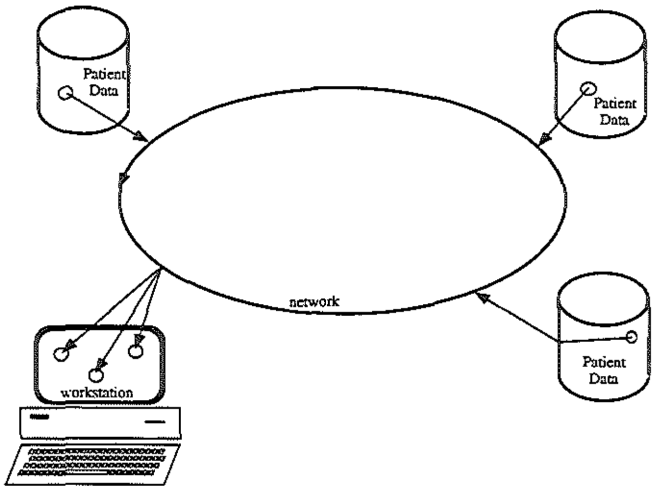


Figure 1. Overview of typical computer organization in a hospital.

One way to integrate all the sources into one computer-based medical record is to transfer all data directly after collection to one fairly large central database that contains all data of patients. Requests to see a patient's medical record are taken care of by this central database. A second approach postpones the integration of data until the patient's medical record is requested for inspection. In this approach, a workstation is often used as the place where the data is combined to one computer-based medical record. The first approach is the traditional one, followed by most hospital information systems and, if succeeded, integration is then relatively easy to achieve. However, it lacks the flexibility of the second approach. The second approach allows users to add to the set of databases that together form the computer-based medical record, newly developed databases with specific data, e.g., collected for research problems. These new databases can be added without changing the other databases. Moreover, the second approach allows for integration of databases with images and biosignals, as well as medical equipment. Powerful personal computers can be used to present this medical record in a more user-friendly way, together with the presentation of images and signals. These computers are often equipped with large displays that can show images and signals (graphics) together with text. Such graphical computers can also be used to provide a more friendly access to databases and programs, and are often called graphical workstations.

Similarly, programs to interpret ECGs, to search medical literature databases, to provide medical decision support, etc., can be linked to the computer-based medical record. Since these programs are not necessarily operational on a central system, but are running on a host of different computers such as the ones offered by industry, this allows clinicians to take advantage of the newest developments and to use their own preferred programs for processing (for instance, a specific text editor).

Why is integration so difficult?
When integrating different databases and programs, one is confronted with the following problems.

Firstly, to access data in a database, one has to know the commands that should be entered to extract data from the database. In general, databases from different vendors use different commands to retrieve the data, and their organization and representation (= data format) is different for each database. Integration of various databases requires methods that support all these different commands to the different databases and that access the data in different data formats.

Secondly, various databases often reside on different computers. A connection between these computers, a network, can be used to transport data between these computers. Many different networks exist, each with their own commands and data formats to transport the data. Consequently, integration also includes the ability to cope with (different) network data transport between computers.

Thirdly, the commands to start a program and to select a function within a program vary from program to program. Not only the commands themselves, but also the syntaxes of the command languages show great differences. Data used by some program has to be provided in a particular data format and, again, these data formats differ between programs.

Evidently, integration is difficult because of the diversity of commands, command languages and data formats. The distribution of databases and programs over different computers can be solved by using a network that connects these computers with each other. Again, this adds to the complexity of the problem of integration.

Is integration still worthwhile?
Why bothering to solve integration problems if a central system with all programs included solves all problems? As already mentioned, a central system is not as flexible as an integrated system; new databases, new programs, new computers can be integrated using the second approach without having to install a completely new system. Moreover, data can be stored at places where the data logically belong: typical cardiological data are stored in the computer of the cardiology department, laboratory data in the computer of the laboratory, etc. If authorized, data from all these (departmental) databases can be combined into one integrated computer-based medical record. Furthermore, software industries deliver a host of different systems (for the CCU, the catheterization laboratory, cardiac surgery, etc.) that cannot run on one central system.

Integration is also required because the data are used for many different purposes. Besides for patient care, data can also be used for clinical data analysis, quality assessment, education, etc. For instance, a cardiologist may use a dataset for further analysis and may apply statistical programs and presentation programs running on different computers.

Integration of commercially available software reduces the development and maintenance of local software, and enables hospital computer groups to give more support to the specific medical programs and databases.

Last but not least, the price-performance ratio of a decentralized system is much higher than that of a centralized system; a collection of small computers providing a particular amount of processing power is much cheaper than one machine with the same amount of processing power. Especially when one wants to use the graphical capabilities of workstations or PCs, a decentralized system is much more adequate.

The HERMES integrated workstation

Since 1987, experience with integration has been obtained from our software integration project. This resulted in the development of a prototype integrated workstation for support of clinical data analysis. The prototype integrates various databases with statistical programs (e.g., BMDP [4]), graphical presentation programs such as Harvard Graphics, and a statistical pattern recognition system (ISPAHAN [5]). In addition, programs to display images (CTs, scintigrams, ultrasound images) and biosignals (ECGs, Holter signals, monitoring data) have been included. Insight in the integration problems has been obtained, and various approaches have been tested as to their feasibility. The prototype workstation was extensively evaluated by users [6].

One of the main research questions has been the development of a general method to combine flexibility and extensibility with minimal software maintenance; it should be possible to have new databases and new programs easily integrated into the HERMES environment without having to rewrite the complete system, and to provide mechanisms to have new program releases easily installed. This methodology, often called "plug-and-play", requires a different organization of access to databases and programs than normally done; our HERMES project is aiming at such an approach [7]. The prototype phase was successfully completed, and in cooperation with other partners the next generation of HERMES is now being developed.

Related research

Integration is not only a problem for the medical discipline. Office automation, computer-aided design and manufacturing (CAD/CAM [8]) are all facing the same problems. However, the solutions in these fields tend to be more directed at rewriting the programs and databases that have to be integrated rather than leaving them untouched. In the medical field, the mass of different databases and programs is so large, that such a solution cannot be followed.

The research projects in medical informatics on this topic can be roughly divided into three main streams:

(1) *To start new development of databases and programs that can be integrated.*
This approach starts with the definition of the data format and command language. According to this specification, new databases and programs are developed. This approach does in principle not use existing databases or programs and assumes hospitals to start all over again. An example of this approach is HELIOS [9].

(2) *Proper selection of databases and programs that have similar command languages and data formats.*
If one selects databases and programs with respect to the presence of already available links and uniformity, one can reduce the integration problems and, at the other hand, still use commercially available products. The DeSygner workstation of Greenes [10,11] is an example of a proper selection of databases and programs for medical decision support. However, for many tasks, this approach cannot be followed due to the fact that most of the time there is already an installed base of databases, programs and medical equipment.

(3) *To use existing databases and programs and develop an integration methodology that links these databases and programs in a workstation.*
This approach has as starting point already existing databases and programs, and defines integration layers around them that handle the differences. This approach is restricted in the sense that (1) not all programs can be integrated, (2) that the integration is sometimes not very efficient, and (3) it sometimes only covers part of the program or database functionality. Examples of this approach are the

Integrated Academic Information Management System project (IAIMS) of the National Library of Medicine in the USA [12] and the HERMES project [13]. The IAIMS project initially focuses more on using existing databases only, whereas the HERMES project aims at integrating programs as well. Hewlett Packard's Physicians Workstation project lies in between (2) and (3). It integrates an existing database with newly developed programs [14].

Database integration in the HERMES prototype

Current clinical databases are not very well suitable for support of clinical research: firstly, the storage and retrieval is optimized for one patient and it lacks methods for searching the databases for patients satisfying a particular search criterion, and retrieving data for large sets of patients is not supported. Secondly, keeping a dataset stable for the period of the clinical research cannot be required. For the HERMES prototype, the decision has been made to have data from clinical research databases copied into a research database. Searches are applied to that research database only and no changes are allowed during the course of the research project. This approach has also as benefit that patient data can be made anonymous, i.e., there is no possibility to deduce the data to a particular patient. In the HERMES prototype, the following databases (not all from Cardiology) have been obtained from clinical information systems, in order to test the flexibility of dealing with different databases and data formats:

(1) Import of data from a hospital information system (BAZIS database). Data were obtained from the andrology department, including laboratory data.

(2) Data from dBaseIII:
   (a) A database that contains data about Somatostatine scans.
   (b) A database from the Thoraxcentre with data about the follow-up of hypertrophy patients.

(3) Import of data from a departmental information system (the Thoraxcentre TUS database), containing follow-up data for patients with an myocardial infarction.

(4) Import from a statistical database.
   (a) SAS: data about occurrence of myocard infarcts of students.
   (b) SPSS: follow-up data about colon-cancer patients.

(5) Import of data from a primary care information system:
data about drug prescriptions for general practitioners, collected in the general

practitioner information system ELIAS, e.g., containing data on hypertension treatment.

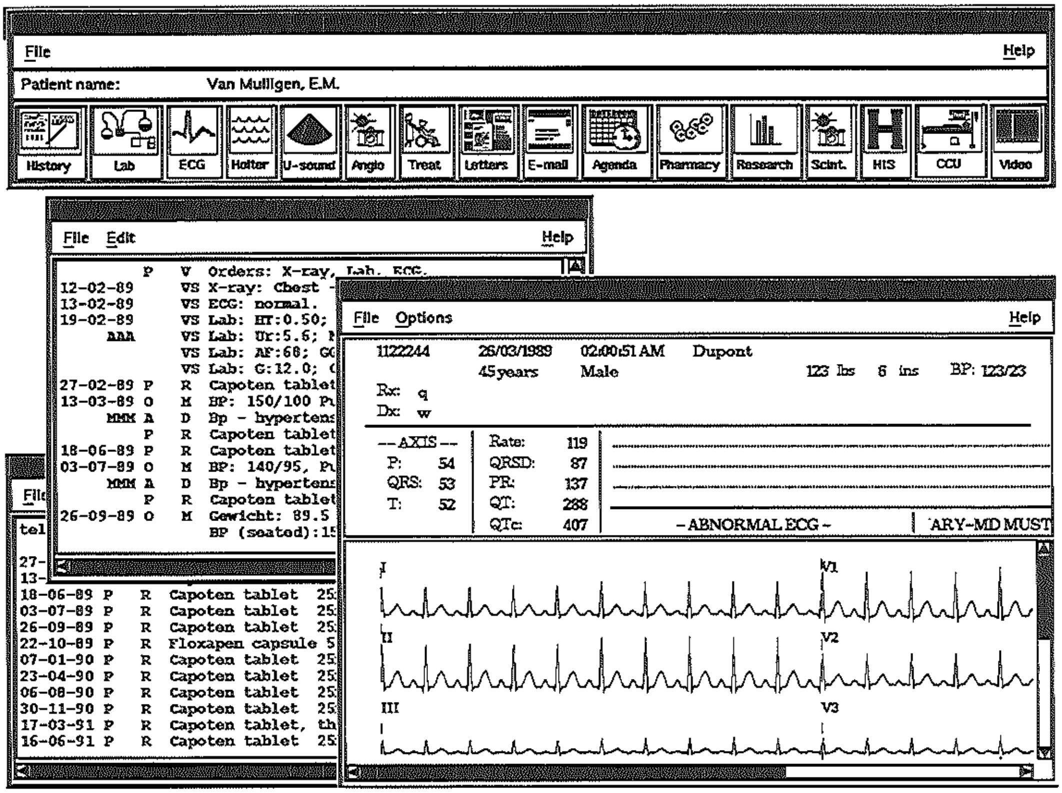
Data from these databases can be used directly in statistical programs and graphical presentation programs. However, combining data from different databases is not supported in the prototype, but is currently being developed in the next generation of the HERMES workstation.

Patient care

The concept of integration was explained with the above examples of databases. Likewise, various kinds of data, such as medical images and signals, can be integrated for the support of patient care. In clinical practice this is an obvious and much needed facility. A prototype of such a facility has been developed within the prototype HERMES workstation. This program anticipates the integration of different equipment from different vendors (ECG recording equipment, Echo systems, Angio systems, CT scanners, etc.) now used for patient care. Within the patient care facility of HERMES, all data from such equipment can be presented to the user (see Figure 2). These sources of data are represented by self-explaining icons. In the next generation of HERMES, the network links with these databases will be established and the functionality of this patient care program will be further extended.

Medical data model

One of the lessons learned from the HERMES prototype is the importance of a medical data model. A medical data model describes all data that are somewhere present in a database. This data model contains descriptive information per data element, i.e., its type, its valid range, its pretty name, synonyms, valid codes, etc. This data model can be used to formulate retrieval commands for multi-databases. Besides, the model can be used by programs for all kinds of purposes: entering data can be controlled with information from the data model, graphical presentation programs can use the pretty names for presentation, statistical programs can use the type and validity ranges for selecting the statistical model, etc. In the HERMES project, a knowledge editor [15] has been developed that supports such a medical data model. The knowledge editor supports the definition of medical relations between data in the databases (see Figure 3).



Patient name: Van Mulligen, E.M.

```
         P    V  Orders: X-ray, Lab, ECG.
12-02-89      VS X-ray: Chest -
13-02-89      VS ECG: normal.
19-02-89      VS Lab: HT:0.50;
      AAA     VS Lab: Ur:5.6; 
              VS Lab: AF:68; GC
              VS Lab: G:12.0; (
27-02-89 P    R  Capoten tablet
13-03-89 O    M  BP: 150/100 Pu
      MMM A   D  Bp - hypertens
         P    R  Capoten tablet
18-06-89 P    R  Capoten tablet
03-07-89 O    M  BP: 140/95, Pu
      MMM A   D  Bp - hypertens
         P    R  Capoten tablet
26-09-89 O    M  Gewicht: 89.5
                 BP (seated):15
```

```
tel
27-
13-
18-06-89 P   R  Capoten tablet   25
03-07-89 P   R  Capoten tablet   25
26-09-89 P   R  Capoten tablet   25
22-10-89 P   R  Floxapen capsule 5
07-01-90 P   R  Capoten tablet   25
23-04-90 P   R  Capoten tablet   25
06-08-90 P   R  Capoten tablet   25
30-11-90 P   R  Capoten tablet   25
17-03-91 P   R  Capoten tablet, th
16-06-91 P   R  Capoten tablet   25
```

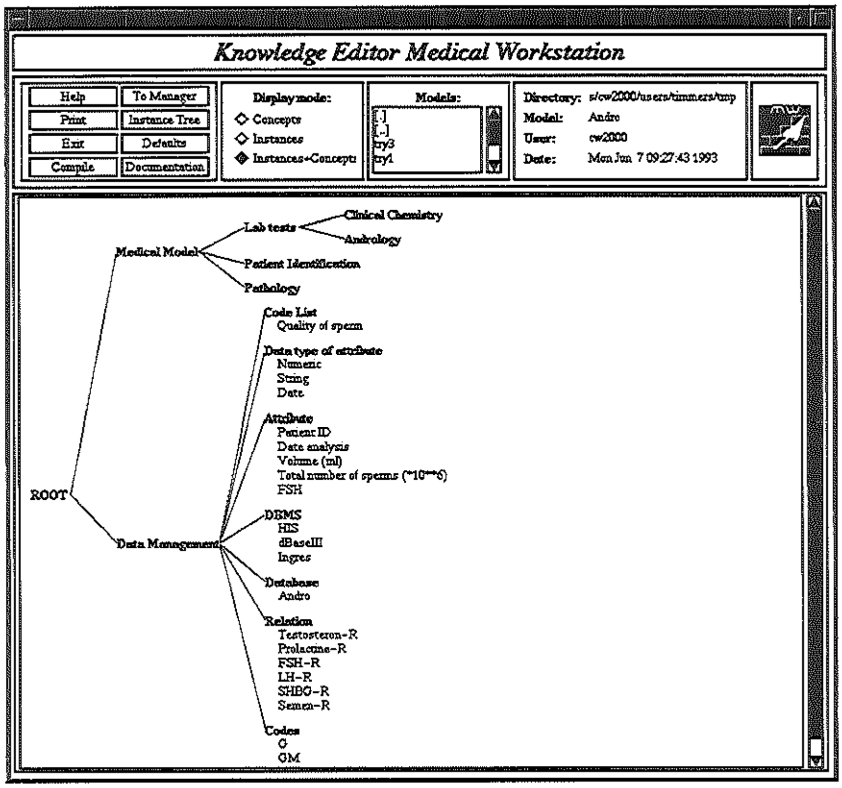
```
1122244      26/03/1989    02:00:51 AM    Dupont
             45years       Male                 123 lbs   6 ins   BP: 123/23
Rx:  q
Dx:  w

--AXIS--    Rate:    119
 P:    54   QRSD:     87
 QRS:  53   PR:      137
 T:    52   QT:      288
            QTc:     407      - ABNORMAL ECG -      'ARY-MD MUST
```

Figure 2. Presentation of data and programs at the HERMES prototype.

User friendliness

The HERMES integrated workstation offers users a friendly access to all databases and programs. This friendliness can be seen at a number of levels. First of all, all databases and programs can be reached from any one workstation. Secondly, HERMES hides from the user all the differences that exist between data formats, command languages and network communication; the user does not have to remember all these differences. Thirdly, data are automatically translated to the data format of the program selected by the user. The Hermes prototype covers most integration problems that may exist in clinical practice[1].



Display mode:
◇ Concepts
◇ Instances
[✓ Instances+Concepts]

Models:
[.]
[..]
try3
try1

Directory: s/cw2000/users/timmers/tmp
Model: Andro
User: cw2000
Date: Mon Jun 7 09:27:43 1993

Figure 3. Part of a data-model with relationships between the data attributes.

In addition, the medical data model is used to help users with various programs that need information about the data or information about the relationships between medical data.

Conclusions

Integrated workstations can be built with existing databases and programs rather than having to rewrite them. A flexible and extensible architecture has been developed to achieve this. The integration methodology limits the dependence on the commands and data formats of the integrated databases and programs. Our methodology supports the integration of data from different medical equipment for the support of patient care.

A medical data model is essential for formulating commands for multiple databases and for entering correct data about the patient. This data model will be the core of the next generation of HERMES.

The next generation of HERMES will include network links with the hospital information system (BAZIS) and the information system of the Thoraxcentre (TUS). Data from these systems will be directly available for clinical research and for integration in the computer-based medical record at the clinician's desk. The facilities for clinical data analysis will be extended in this next generation, as well as the facilities for displaying images and biosignals.

---

[1]The prototype runs on Hewlett Packard HP9000 series workstations and on 80386/486 PCs running SCO-UNIX. Both run with UNIX system V and uses the OSF/Motif X11 window system for the graphical user-interface. For database queries the standard language SQL has been used. Arpa/Berkeley network software is used to communicate through the Ethernet running the TCP/IP protocol with other computers.

References

[1] Timmers T, Van Mulligen E M, Van den Heuvel F, Van Bemmel J H. MW2000 - a workstation for the support of clinical research and patient care in cardiology. The Thoraxcentre journal, 3(4);1991:8-11

[2] Blois, M.S. Information and medicine. University of California Press, Berkeley. 1984.

[3] Blum. B.I. The dynamics of a clinical information system. In: *Proceedings of the Seventh World Congress on Medical Informatics*. Geneva. Amsterdam: North-Holland Publishing Company. 1992:168-73.

[4] Dixon, W.J. BMDP statistical software manual. University of California Press, Berkeley, 1981.

[5] Gelsema E S. ISPAHAN: an interactive system for pattern analysis: structure and capabilities. In: Gelsema E S, Kanal L N, eds. *Pattern Recognition in Practice*. 1980. Amsterdam: North-Holland Publishing Company. 1980:481-91.

[6] Van Mulligen E M, Timmers T, Van Bemmel J H. Evaluation of a Medical Workstation. (submitted for publication).

[7] Van Mulligen E M, Timmers T, Van Bemmel J H and Van den Heuvel F. functional requirements for an integrated medical workstation. In: *Proceedings of the Seventh World Congress on Medical Informatics*. Geneva. Amsterdam: North-Holland Publishing Company. 1992:1261-6.

[8] Hsu C, Skevington C. Integration of data and knowledge in manufacturing enterprises: a conceptual framework. Journal of Manufacturing Systems 1986;6:277-85.

[9] Degoulet P, Coignard F C J, et al. The HELIOS european project on software engineering. In: Timmers T, Blum B.I., eds. *Proceedings of the IMIA Working Conference on Software Engineering in Medical Informatics*. Amsterdam. Amsterdam: North-Holland Publishing Company. 1990:125-37.

[10] Greenes R A. Promoting productivity by propagating the practice of "plug-compatible" programming. In: *Proceedings of the 14th Symposium on Computer Applications in Medical Care*. Washington DC. New York: IEEE Computer Society Press. 1990:22-6.

[11] Greenes R A. "Locators", "Constructors" and "Trackers": meta-level tools for supporting health-care professional information needs in a distributed computing milieu. In: *Proceeedings of the Seventh World Congress on Medical Informatics*. Geneva. Amsterdam: North-Holland Publishing Company. 1992:2-7.

[12] Lunin L F, Ball M J. Perspectives on Integrated Academic Information Management System (IAIMS). In: *Journal of the American Society for Information Science*, 39, 3 (1988) p102-12.

[13] Van Mulligen E M, Timmers T and Van den Heuvel F. A framework for uniform access to data, software and knowledge. In: *Proceedings of the 15th Annual Symposium on Computer Applications in Medical Care*. Washington DC. New York: McGraw-Hill. 1991:496-500.

[14] Tang P C, Annevelink J, et al. Physicians' workstations: integrated management for clinicians. In: *Proceedings of the 15th Annual Symposium on Computer Applications in Medical Care*. Washington DC. New York: McGraw-Hill. 1991:569-73.

[15] Van den Heuvel F, Timmers T, Van Mulligen E M, Hess J. Knowledge-based modeling for the classification and follow-up of patients with congenital heart disease. In: *Proceedings of Computers in Cardiology Conference*. Venice. Los Alamitos: IEEE Computer Society Press. 1991:737-40.

CHAPTER 4

User Evaluation Of An Integrated Medical Workstation For Clinical Data Analysis

E.M. van Mulligen[1,2], T. Timmers[1], J.H. van Bemmel[1]

[1]Dept of Medical Informatics, Erasmus University Rotterdam, The Netherlands
[2]University Hospital Dijkzigt, Rotterdam, The Netherlands

Accepted for Methods of Information in Medicine

Abstract

Results are presented of the user evaluation of an integrated medical workstation for support of clinical research. Twenty-seven users were recruited from the medical and scientific staff of the University Hospital Dijkzigt, the Faculty of Medicine of the Erasmus University Rotterdam, and from medical institutions outside the Erasmus University and all were given a written, self-contained tutorial. Subsequently, an experiment was done in which six clinical data analysis problems had to be solved and an evaluation form was filled out. The aim of this user evaluation was to obtain insight in the benefits of integration for support of clinical data analysis for clinicians and biomedical researchers. The problems were divided into two sets, with gradually more complex problems. In the first set users were guided in a stepwise fashion to solve the problems. In the second set each stepwise problem had an open counterpart. During the evaluation, the workstation continuously recorded the user's actions. From these results significant differences became apparent between clinicians and non-clinicians for the correctness (means 54% and 81%, respectively, *p*=0.04), completeness (means 64% and 88%, respectively, *p*=0.01), and number of problems solved (means 67% and 90%, respectively, *p*=0.01). These differences were absent for the stepwise problems. Clinicians tend to skip more problems than biomedical researchers. No statistically significant differences were found between users with and without clinical data analysis experience, for correctness (means 74% and 72%, respectively, *p*=0.95), and completeness (means 82% and 79%, respectively, *p*=0.40). It appeared that various clinical research problems can be solved easily with support of the workstation; the results of this experiment can be used as guidance for the development of the successor of this prototype workstation and serve as a reference for the assessment of next versions.

Keywords: Integration, Evaluation, Workstation

1. Introduction

In the last decade, powerful graphical workstations and network technology were among the main developments in the computer industry. The fall of hardware prices and the standardization efforts at the network level enable the introduction of the personal computer (PC) at each clinician's desk. Despite their processing power, PCs are often only used as a terminal window on applications residing on a central host and for text processing. The computer experience of the clinician determines to a large extent whether different applications either on a PC or on the central computer are used.

The problem addressed by the HERMES[*] integrated workstation is that there exists a wide range of applications with different user-interfaces, data formats and command languages, located on various hosts with different hardware and operating systems. The approach taken to solve this problem is the one of encapsulating the applications with a software layer that provides a uniform network access to those applications [1,2]. In addition, automatic data translation and command generation is provided by these encapsulators. In fact, the client-server paradigm with open distributed processing has been followed [3]. We implemented the encapsulators as network services that can be accessed from the workstation's graphical user-interface, which incidentally is a client itself.

Although the architecture is of a general nature, we developed the workstation for the support of both clinical data analysis and patient care by clinicians. From this follows our particular choice of applications and design of the user-interface. However, we believe that the organization of the assessment study as presented in this paper is not restricted to our target domain or our approach to workstation integration.

To the clinical user, the user-interface of the HERMES workstation gives access to all functions that are provided by the applications on a point-and-click basis. For example, after identifying data from a general data model, a statistical function can be requested from a pulldown menu and the results are shown graphically in a separate window. Functions of the following applications have been integrated in HERMES: access to the database management systems INGRES and dBaseIII, to a hospital information system and a departmental information system, to the statistical functions of BMDP and locally

[*)] HERMES is an acronym for HEalth Research MEdical System.

developed statistical applications, and to the graphical presentation functions of HarvardGraphics and the X11-environment.

Our approach of leaving databases and applications unchanged by encapsulation and combining data on demand has some advantages as compared with building one large integrated database [4]: (1) data can be stored at places that are best suited (e.g., close to where the data are collected and used), (2) commercially available applications can be used without modifying them. This latter feature was recognized by Andrews and Power as an important new approach [5,6] that is consistent with the ISO distributed processing model [7].

Although HERMES has been initially developed for support of clinical data analysis, its architecture has also been applied for the support of patient care. With clinical data analysis, data of many patients are transferred between the applications and the diversity between the applications is much larger. The problems tackled for clinical data analysis are in fact a superset of those of integrating databases and applications for patient care.

User evaluation

Although much research has been devoted to developing methods for evaluation of applications [8,9], these methods cannot be applied directly to our situation. First of all, we do not intend to evaluate the applications themselves, but rather to evaluate whether clinicians could independently solve their clinical data analysis problems with an integrated medical workstation. Secondly, no quantitative comparison could be made with the current support of clinical data analysis for clinicians; most clinicians can rely on a statistician and/or a database manager for solving their problems. Therefore, we included biomedical researchers in the study as a reference for comparison with the results of clinicians.

The formal, quantitative evaluation was motivated by the fact that for most clinical users this was the first time that they solved a clinical data analysis problem at their own. The lack of a comparison makes qualitative remarks difficult to understand. Secondly, sometimes the clinician was not aware of making errors and still judged the workstation to be of excellent help. We included only two qualitative statements that clinicians could make about the expected time savings of HERMES (as compared with the tedious path they have now) and whether they prefer HERMES to the current situation of support (e.g., by a statistician or a database manager).

Six clinical data analysis problems were presented to the users and all interactions with HERMES were logged. From these logging data, evaluation parameters have been derived and a list of errors has been compiled. A profile of the errors can be used to adjust the design of HERMES so that these errors will not occur anyway.

In this paper, the assessment study and its results are described. In the next section we describe the organization of the assessment study. The results section presents the statistical evaluation of the measurements obtained from the logfile. Finally, we discuss the implications and results of this assessment study for future developments of integrated medical workstations. Background information on HERMES has been published elsewhere; see [1,2] and the references therein.

2. Material and Methods

The evaluation by clinicians was intended to assess the possibility of the workstation in providing enough support for representative, though not very complex clinical research problems. The seamless integration of applications used for clinical research can also be helpful for biomedical researchers, such as statisticians. Therefore, together with clinicians they were invited to participate in the user evaluation to test the workstation for its ability to solve clinical research problems.

The clinicians were recruited by personal invitation, as much as possible from different medical specialties. Most clinicians were in some way already involved in clinical research, albeit not necessarily working themselves with computers. The biomedical researchers were invited to evaluate whether they regarded the HERMES integration concept positively since they are already used to solve their problems with the current, non-integrated computer environments.

In addition to measuring the differences between clinicians (Cl) and biomedical researchers (NCl), we also looked at differences between users with and without clinical data analysis experience (E and NE, respectively). The division of users in experienced and non-experienced just on the basis of what they stated themselves seemed to be rather subjective. Therefore, we used a more objective division in experienced versus non-experienced on the basis of the number of publications indexed in MedLine from 1987-1992. No book reviews, letters to the editor, or other secondary publications were taken into account in the search. If a publication as a first author was found in this period, or one or more publications per year as co-author, the user was assigned to the group of experienced users (E).

2.1 Introductory Course and Experiment

A written, self-contained tutorial lasting about two hours introduced the principles of the workstation and the interaction with some of the most frequently used modules: data selection, data inspection, cross tables, p-values, and a graphical presentation module. The course ended with two typical clinical research problems which were presented in a step-by-step fashion guided by the tutorial. Finally, two other problems were included that users had to solve alone, i.e., without guidance, in order to test whether users were able to use the workstation unaided. During the course, interaction with the supervisor of the evaluation study was restricted to situations where the user was not able to continue; users were asked to solve the problems as much as possible independently.

After the course, the assessment study was done with six clinical research problems using a database of patients with colon cancer. The data about patient identification, surgeries, consultations, location and staging of cancer, other types of cancer, and survival were collected and studied at the Leiden University Hospital by Bloem et al. [10]. One of the goals of this research database was to investigate the relation between the progression of cancer (the so-called Dukes stages), gender, age, and duration of symptoms on the one hand, and period of survival and mortality on the other hand. The data in this database were completed and a medical dictionary that contained the mapping between medical terms and database attributes, relations, and codes was available on the workstation. Users could select data from this database by pointing with a mouse at the terms of the dictionary and by specifying search criteria by using an interactive query program. In our departmental setting, both the cancer database and the statistical functions were remotely accessed from the workstation through the network.

The problems to be solved were presented on paper and no interaction with the supervisor was allowed (see Appendix A for research problems). The six problems (in Table 2 numbered I-VI), dealing with data from the colon cancer database, were arranged into two groups of three problems each. The first group (I-III) was organized in a step-by-step mode: all individual subproblems to be taken were listed and users had to solve all steps sequentially before proceeding. The second group (IV-VI) consisted of open questions: no information was given about the data and the statistical tests necessary to solve the problem. Each problem in the first group had a counterpart in the second group regarding complexity. All three problems had a different level of complexity: the first problems had an unconditional data selection combined with descriptive procedures (e.g., cross tables or histograms). The second problem in each set required a repeated conditional selection in conjunction with more advanced statistics, such as chi-square tests and cross-table manipulation functions. The third problem in each set was included to assess the user's response to a test not presented in the introductory course. It was assumed that confrontation with survival analysis would not be a problem when users were accustomed to the interface-style of the workstation. Problems II, IV and V consisted of an A and a B part (e.g., the A part related to males and the B part to females).

After the assessment study, users completed an evaluation form to indicate their general experience with clinical research, the frequency of their solving research problems, the scope of their research, and their impression of the workstation: whether it saved time, how it compared with current processing support, etc.

Workstation Expansion
To get an impression about the support provided by the workstation, the actions that the user should carry out for problem I are shown in Table 1 together with the actions the workstation actually performs hidden for the user. These actions include translations of data, transmissions through the network, and generation of commands, together with starting applications.

2.2 Measurements
In the assessment study, all user actions were logged in a logfile. This included all events that caused something to happen in the workstation. For each event, the time of occurrence and the time of completion were recorded. Two buttons on the screen were included for the experiment to stop and resume the logging clock, e.g., for coffee breaks.

Table 1. User actions (left hand column) as expanded and carried out by the HERMES prototype workstation (right hand column). The whole sequence is a typical example of how data can be selected from the database, inspected, displayed as a cross table, and printed.

User request	Hidden workstation operations
Select medical terms Age, Gender, Dukes stages from a list.	• Translate the selected attributes to the relation attribute names of the database.
Specify search condition, e.g., Age>50.	• Translate the selection and the condition to a correct DBMS query. (The workstation knows per medical term what values or codes are allowed.)
Give command to execute task.	• Send the query through the network to the computer that holds the database. • Start the query interpreter of the DBMS. • Retrieve the outcome through the network to the user's workstation. • Translate the data to a common format.
Inspect results.	• Translate the data to the inspection module format. • Start the graphical inspection program.
Save results.	• Save the data in the workstation's Intermediate Storage Format (ISF), together with information about the query and the format of the data.
Construct cross-table.	• Send the data to the computer that holds the cross-table module. • Translate data to the format of the cross-table module. • Start the computation of the cross-table module. • Translate the results to the ISF format. • Retrieve results through the network to the user's workstation.
Print results.	• Generate a postscript output of the graphs/tables. • Send the output to a postscript printer.

These logging data could be inspected later to assess the correctness of answers and to determine the time spent in working with specific modules by summation of the time of individual steps. The logging data of the experiments were compared with logging data of a correct solution that did not include any unnecessary steps. This solution was verified against an optimal sequence of interaction steps that was determined by the creator of HERMES and the results were compared with those presented by Bloem et al. [10].

2.3 Users

Twenty-seven users (13 biomedical researchers and 14 clinicians) participated voluntarily in the evaluation; one clinician followed the course but did not finalize the experiments. All clinicians were recruited from clinical departments of the University Hospital Dijkzigt and most of them were involved in some clinical research project and were not totally inexperienced with computers. The biomedical researchers were recruited from the Faculty of Medicine of the Erasmus University Rotterdam, from clinical departments of the University Hospital Dijkzigt, and from medical institutions outside the Erasmus University.

3. Results

All 27 users completed the course and answered correctly the problems in the final test at the end of the introductory course. None of the users had previous experience with the workstation. Table 2 presents the most pertinent data collected during the assessment study, together with the results of the problem solving, subjective assessment data, and the number of publications indexed in MedLine during 1987-92.

Column Cl shows whether the user is a clinician (involved in patient care) or not (+ or -, respectively). The user's self-assessment of his or her own estimated degree of experience with clinical data analysis is presented in column Cld. Code 5 indicates expert level, 4 much, 3 normal, 2 little, and 1 no experience. The total number of publications indexed in MedLine over 1987-92, P_n, is given for each user. This was also done for publications in which the user was the first author (P_1). The distribution of the time for each of the six clinical data analysis problems (T_c) is shown in Figure 1, for experienced (E) and non-experienced (NE) users. Each user was requested, at the end of the study, to give his/her opinion about the usability of the workstation, presented in columns Cp and Tsa. Cp contains a subjective comparison of the workstation with current support for clinical data analysis: code 3 indicates that the workstation is better, 2 that it performs equally well, and 1 that it is less supportive. Tsa contains a subjective estimation of the time saving of the workstation using the same codes as for Cp.

Table 2. Raw data, as collected during the assessment and as obtained from an evaluation form and MedLine.

Cl	I	II$_a$	II$_b$	III	IV$_a$	IV$_b$	V$_a$	V$_b$	VI	Cld	Cp	Tsa	P$_1$	P$_n$
+	4	4	4	4	4	4	4	4	4	3	3	3	1	9
+	4	4	4	4	4	4	4	1	4	3	3	3	20	81
+	4	4	4	4	4	4	3	1	4	4	3	2	5	19
+	4	4	4	4	4	4	3	-	-	4	3	2	8	14
+	3	3	1	4	4	1	3	-	-	4	3	2	5	13
+	3	2	-	-	-	-	-	-	-	4	3	2	6	19
+	4	4	4	4	4	4	3	-	-	3	3	2	0	3
+	4	2	-	-	-	-	-	-	-	3	3	2	0	1
-	4	4	4	4	4	4	4	4	4	1	2	1	3	4
-	4	4	4	4	4	4	4	4	4	4	2	3	1	23
-	4	4	4	4	4	4	4	1	4	2	3	2	5	14
-	4	4	3	4	4	4	4	1	4	4	2	2	13	43
-	4	4	4	4	4	4	4	4	4	1	3	2	6	10
-	4	4	4	4	4	1	4	4	4	3	3	3	4	11
-	4	4	1	4	4	3	3	1	4	4	2	1	0	56
-	3	4	4	4	4	4	4	1	4	3	3	3	0	1
-	3	2	3	2	-	-	-	-	-	4	3	3	0	1
-	4	4	4	4	4	4	4	4	4	4	2	2	0	0
-	4	4	4	4	4	4	4	4	4	3	3	3	0	0
-	4	4	4	4	4	4	4	4	4	2	2	2	0	0
-	4	4	3	4	4	4	4	4	4	1	2	2	0	0
-	4	3	4	4	4	4	4	4	4	4	1	1	0	0
-	4	4	4	4	4	4	4	4	4	2	3	2	0	0
-	4	3	4	4	4	4	4	4	4	2	3	3	0	0
-	4	4	3	4	4	4	4	4	4	3	3	3	0	0
-	4	3	1	4	-	-	-	-	-	3	3	3	0	0
-	-	-	-	-	-	-	-	-	-	1	2	3	0	0

Cl	= clincian, directly involved in patient care,
I-VI	= status per question
	(4=correct completed,3=incorrect completed,2=incorrect and incomplete,1=not done,-=quitted)
	($_a$ and $_b$ parts are equivalent questions for different patient selections),
Cld	= clinical data analysis experience (5=expert,4=much,3=normal,2=little,1=none),
Cp	= subjective comparison with current support for clinical data analysis (3=better,2=equal,1=worse),
Tsa	= subjective estimation of time savings of workstation approach (3=better,2=equal,1=worse),
P$_1$	= number of publications MedLine 1987-1992, if first author,
P$_n$	= number of publications MedLine 1987-1992, if author.

The results of the six problems to be solved by the users are provided in columns I to VI. Three problems contained an A and a B part. The B part is a repetition of the A part, but for a different group of patients, e.g., females instead of males. The type of data analysis of a B part, however, is identical to that of an A part. Answers to the problems have been scored both for correctness and completeness. The codes provided in columns I-VI represent five possible outcomes: 4 indicates a correct and complete answer, 3 an incorrect but complete answer, 2 an incorrect and incomplete answer, 1 indicates that the user skipped the problem, and a "-" that a user had already stopped before he could start that problem. Note that the last user did not tackle any problem, although the tutorial was completed correctly; these results were excluded from further analysis.

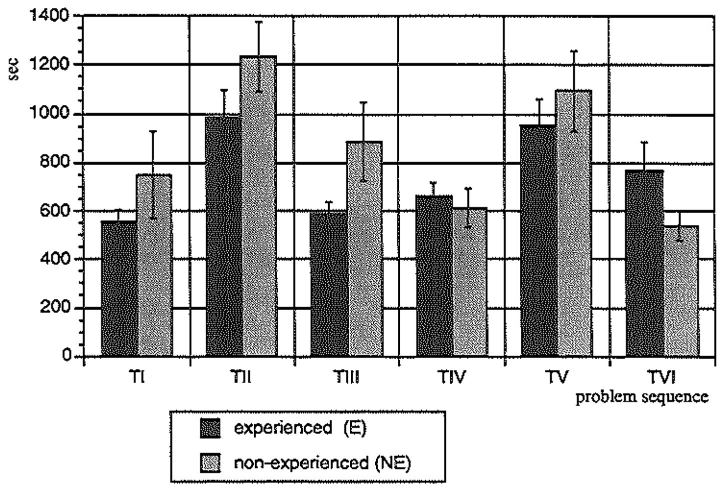


Figure 1. Histogram of the mean time (with standard error of the mean) per problem for experienced and non-experienced users. A division is made according to whether users are experienced (E) or non-experienced (NE) in clinical data analysis. T_1 is the time used for problem I, etc.

3.1 Analysis

For the two different groupings of users, i.e., Cl versus NCl, and E versus NE, the results are shown in Table 3, where mean values are shown for the different groups. Score S_1 is computed as the fraction of all correctly and completely solved problems; S_2 includes S_1 and all problems that were not correctly solved but completed; S_3 includes S_2 and all problems that were solved but were not correct and not complete; S_4 contains the fraction of all problems that were not tackled. For the three step-wise problems (I-III), the correctness score S_1 is shown in column $S_{1\alpha}$ and for the last three problems (IV-VI) in $S_{1\beta}$.

Table 3. Summary table for results derived from the initial raw data.

	#	S_1	S_2	S_3	$S_{1\alpha}$	$S_{1\beta}$	S_4
Cl	8	0.54	0.64	0.67	0.69	0.43	0.33
NCl	18	0.81	0.88	0.90	0.82	0.81	0.10
E	13	0.74	0.82	0.83	0.82	0.68	0.17
NE	13	0.72	0.79	0.82	0.73	0.71	0.18

Cl/NCl	=	involved in patient care/not involved
E/NE	=	research experience as derived from MedLine/no experience
#	=	number of users
S_1	=	mean fraction of correct and complete questions (questions with code=4),
S_2	=	mean fraction of complete questions (questions with code=4 or 3),
S_3	=	mean fraction of answered questions (questions with code=4, 3 or 2),
$S_{1\alpha}$	=	mean fraction of correct and complete questions in I-III (questions with code 4),
$S_{1\beta}$	=	mean fraction of correct and complete questions in IV-VI (questions with code 4),
S_4	=	mean fraction of unanswered questions (questions with code=1 or '-')

3.2 Differences

To test the statistical significance of the differences in the answers, the non-parametric Mann-Whitney (Wilcoxon) test has been applied. Statistically significant differences between clinicians and non-clinicians are found for the correctness score S_1 (p=0.04), the completeness score S_2 (p=0.01) and the answered score S_3 (p=0.01). Also the correctness of the open problems $S_{1\beta}$ was different (p=0.01) which did not apply to $S_{1\alpha}$ of the stepwise problems. The fraction of problems not tackled was significantly higher for the clinicians than for the researchers (p=0.01). There is no statistically significant difference in any score between E and NE.

The results for the subjective comparison Cp and time savings Tsa showed for the groups clinicians versus non-clinicians and experienced versus non-experienced no statistically significant differences. The majority of the users (18 out of 27) preferred the workstation to their current working environment (values above 2 indicate a preference for the workstation, less than 2 a preference for the current situation). All 8 clinicians preferred the workstation to their current working environment.

Table 4 presents an overview of the errors users made during the assessment study. The number of the most common errors is presented for clinicians and non-clinicians. Non-clinicians showed considerably less errors than clinicians. Most errors occurred in the data selection, the local saving of a data set and the specification of parameters for creating the survival table. Note that some errors are due to lack of knowledge about the domain, some due to the sequence of operations that the workstation required, and some due to lack of statistical experience.

Table 4. Compilation of errors as registered during the experiments for clinicians (Cl) and non-clinicians (NCl).

Type of Error	Cl		NCl		Total
	N	F_1 (%)	N	F_2 (%)	
Errors in data selection	40	2.6	24	0.7	64
Wrong variables for cross-table	13	0.9	17	0.5	30
More than 2 outcomes for chi-square test	24	1.6	18	0.5	42
No saving of data	45	2.9	71	2.1	116
No table for computing p-values	8	0.5	7	0.2	15
Variables of cross-table interchanged	7	0.5	4	0.1	11
No variables in cross-table selected	22	1.4	26	0.8	48
Wrong parameters specified for survival table	33	2.1	25	0.7	58
Total	192	12.5	192	5.6	384

Cl	=	clinicians, involved in patient care
NCl	=	not involved in patient care
N	=	number of errors
F_1	=	percentage of errors per clinician
F_2	=	percentage of errors per non-clinician

4. Discussion

The first goal of this study was to test the usability of our approach of workstation integration for the support of clinical data analysis for clinicians. The second goal of this evaluation was to establish new design criteria for a next version of HERMES. Although the actual number of users in this evaluation study was small (26 real users), some conclusions can be drawn.

Clinicians and non-clinicians are statistically significantly different for S_1 (correctness), S_2 (completeness), S_3 (problems solved), and S_4 (problems not answered). This was primarily caused by the fact that clinicians skipped 33% of the problems, whereas the non-clinicians only skipped 10%. Clearly, this influenced the correctness and completeness scores of the problems solved. For the fraction of correctly solved problems there was no statistically significant difference (p=0.06) between clinicians and non-clinicians. Apparently, clinicians are equally well able to solve problems with the workstation as non-clinicians. However, from the assessment it seems that they are less inclined to repeat a test once they have seen how it can in principle be solved.

When the scores of the three stepwise problems are compared with those of the three open problems, it can be concluded that there seems to be no significant difference between Cl and NCl for the stepwise problems, but for the three open problems there is a significant difference. Therefore, we conclude that clinicians were supported by a stepwise protocol for solving the problems in this assessment. This finding could have consequences for the design of the successor of the prototype workstation. No significantly different scores are obtained for the groups of experienced and non-experienced users. From the differences between the means of both groups and their corresponding 95% confidence intervals, it is reasonable to expect that these results will also apply when the user groups would have been larger. This finding implies that for the material of our assessment study the workstation was able to "upgrade" the group of non-experienced users to the level of the experienced users.

For the time T_e spent on solving the problems, no statistical difference was obtained between any of the groups, nor for the time spent on the course T_c (on average 120 min.). The 95% confidence intervals for the differences in the means are wide; the values for each of the groups show a considerable dispersion. However, the average time for solving one problem is about 10-13 minutes, which is reasonable. The time necessary to learn and understand the operation of the workstation is about two hours, including the tutorial and explanation of the window-oriented and mouse-controlled environment. It implies that after only a short introductory course users are able to operate the workstation and are further learning by doing.

All groups preferred the workstation above the current support (such as a PC and/or a terminal to a central computer) for clinical data analysis. Possibly, the reasons why clinicians were positive might have been caused by the fact that, in general, they were not used to much computer-supported research by themselves. But also the biomedical researchers, who all have their own computer support environment, preferred on average the workstation approach better than their conventional environment.

From Table 2 it can be seen that there were seven users who clearly did not succeed in or were willing to solve all problems. The reasons why they did not succeed may vary: a few could possibly not solve the problems; or a few did, in spite of the course, still not fully understand the philosophy behind the user-interface and the operation of the workstation, when confronted with open-ended problems; or some possibly stopped because it took them too much time. The real reason for early stopping is difficult to obtain, although it would be interesting to know. We assume that the only user who quit directly after the tutorial without completing any answer, had remaining problems with understanding the user-interface of the workstation. In future assessments, attention should be paid to having more users finalizing the assessment, or at least registering the reason why they quit. Besides, it underlines that the workstation should support users for all new functions with on-the-scene short introductory tutorials; clinicians particularly are used to a very pragmatic approach, such as learning by doing or show and tell.

From the list of errors as depicted in Table 4, an impression can be obtained about the strength of support of HERMES for the various integration aspects. The high error rates for clinicians are caused by only a small number of clinicians. From these figures, it can be seen that selecting the data is one of the aspects that remains difficult. These problems were most frequent when the question consisted of two groups and users had to submit the query twice with different search criteria, but tried to express it as a single query. Another frequent error was the saving of data. In the design of HERMES, we assumed that users had to explicitly request that data were locally saved on the workstation before further processing. Although explained in the introductory tutorial, users forgot to do so and worked with old data. Such errors can only be avoided by both redesigning certain parts of the workstation and by implementing more help-functions and warnings in the workstation, telling the users not to forget certain actions if the workstation detects a possible omission.

The lack of help for statistical modules was most felt when specifying parameters in the statistical user interfaces. For instance, the *p*-value module did not provide information about why no *p*-values could be computed when the table consisted of more than two outcomes. When providing more statistical tests, a future workstation should apparently offer an advisory module to help users in making correct choices among the various possibilities of statistical parameters and tests.

After these considerations, we want to present some concluding remarks:

From the assessment study, it appears that the evaluation of a system such as the prototype workstation with its high user-interaction is very complex. We also realized that references in the literature on quantitative rather than qualitative assessments are very sparse if not totally absent. Yet, we were convinced, and still are, that progress in the further development of these and similar systems can only be made when the evaluation of such systems is done in a quantitative and objective as possible manner. The evaluation of our prototype integrated workstation can serve as a reference model for the assessment of future developments in this domain. Getting insight in the type of errors users make, may help in developing new user-interfaces and desktop metaphors, and user support modules. It allows to repeat this evaluation for new developments and to compare the outcome with the current results.

To our satisfaction the workstation was able to support users who had no experience themselves with clinical data analysis, to the extent that there were no differences between experienced and non-experienced users for correctness and completeness scores. Clinicians had many more problems in solving open problems than non-clinicians; for stepwise problems, no difference was found. Both clinicians and biomedical researchers, and experienced and non-experienced users preferred the workstation to their current processing environment and concluded that it saved time.

Acknowledgements

We gratefully acknowledge the time spent by all participants during this assessment, as well as those who tested the experiments for consistency and errors. Our thanks go to Prof.dr. A. Zwaveling from the University Hospital Leiden for allowing us to use the oncology database and to dr. Th. Stijnen of the Institute of Epidemiology and Statistics of the Erasmus University Rotterdam for providing us with the database and statistical advice. Prof. dr. M. Musen from the section on Medical Informatics, School of Medicine, Stanford University, is acknowledged for his comments on the draft version of this paper. The development of the prototype was supported in part by a grant from the European Programme on Advanced Informatics in Medicine (AIM contract A1043).

References

[1] Van Mulligen EM, Timmers T, De Faria Leao B. Implementation of a medical workstation for research support in cardiology. In: Miller RA, eds. *Proceedings 14th Symposium on Computer Applications in Medical Care.* New York: IEEE Computer Society Press, 1990:769-73.

[2] Van Mulligen EM, Timmers T, Van Bemmel JH. A new architecture for integration of heterogeneous software components. Meth Inform Med, 1993 (to be published).

[3] *Basic Reference Model of Open Distributed Processing.* ISO/IEC JTC1/SC21/WG7 N315. Nederlands Normalisatie-Instituut, December 1990.

[4] Safran C. Using routinely collected data for clinical research. Stat Med 1991;10:559-64.

[5] Power LR. Post-facto integration technology: new discipline for an old practice. In: Ng PA, Ramamoorthy CV, Seifert LC, Yeh RT, eds. *Proceedings First International Conference on Systems Integration.* Los Alamitos: IEEE Computer Society Press, 1990:4-13.

[6] Andrews GR. Paradigm for process interaction in distributed programs. ACM Computing Surveys, 1991; 23:49-90.

[7] Zimmermann H. OSI reference model - the ISO model of architecture for open systems interconnection. IEEE Trans Commun 1980; 26:425-32.

[8] Caschnig J, Klahr Ph, Pople H, Shortliffe EH, Terry A. Evaluation of expert systems: issues and case studies. In: Hayes-Roth F, Waterman DA, Lenat DB, eds. *Building Expert Systems.* New York: Addison-Wesley, 1983:241-79.

[9] Wyatt J, Spiegelhalter D. Evaluating medical decision-aids: what to test, and how? In: Talmon J, Fox J, eds. *System Engineering in Medicine.* Heidelberg: Springer Verlag, 1989:1-13.

[10] Bloem RM, Zwaveling A, Stijnen Th. Adenocarcinoma of the colon and rectum: a report on 624 cases. Netherl J Surgery 1988; 40:121-6.

Appendix A

Problems as presented to the users in the assessment study

Stepwise problems

I. **Determine the distribution of the Dukes stages A, B, C1, C2 and D for men and women above the age of 50 years.**
- Select the oncology database
- Create a new data-set with the variables gender, age and Dukes stage.
- Limit the set to patients with age > 50
- Inspect the data-set
- Create a cross-table for Dukes stage against gender
- Print both the histogram and the table.

II. **Determine for two groups, patients operated between 1958 and 1968 (A) and patients operated between 1968 and 1978 (B) the relation between localization of the tumor and post-surgical death.**
- Create a new data-set with the variables: year of surgery, localization and post-surgical death.
- Restrict year of surgery to the period of 1958 to 1968
- Create a cross-table of localization against post-surgical death.
- Compute a confidence interval by applying 'Contingency Statistics'.
- Select the 99% confidence limits option.
- Print all tables and graphs.
- Repeat these steps for the second group of patients

III. **Determine the survival curves for the different Dukes stages.**
- Create a new data set.
- Select the variables: Dukes stage, survival and status.
- Create a survival table (life table and survival analysis). The time-to-response variable is 'survival', the response condition is: 'status' = 'death'. The unit of the time variable is months.
- Select the 5 Dukes groups in the list of groups.
- Create the curve (Draw) and print (Print) it.

Open problems

IV. What is the relation between the localization of the tumor and the 5 Dukes stages? Determine this for both men (A) and women (B) separately.

V. Is the mortality different for the group that had a surgery at the age between 59 and 68 years (A) and the patients that has a surgery at the age between 69 and 78 years (B)? Determine this for the different Dukes stages with a 90% confidence limit.

VI. Determine the survival curves for men and women with an age less than 60 years. Select patients with Dukes stage B as the group of interest.

CHAPTER 5

Accessors As A Framework
For Integration Of Data And Software

E.M. van Mulligen[1,2], T. Timmers[1], J.H. van Bemmel[1]

[1]Dept of Medical Informatics, Erasmus University Rotterdam, The Netherlands
[2]University Hospital Dijkzigt, Rotterdam, The Netherlands

(to be published)

Abstract

An object-based software integration architecture is described which provides communication within a network between applications according to the client-server model. Each object in this object-based architecture is called an accessor and hides the details for accessing data and functions. This architecture has been designed to also integrate existing applications. A request for data and functions directed to the accessor framework is served by an accessor server that analyzes the request, completes it with essential parameters, and forwards it to the application service object. This application service object hides the access to the data and functions available in an application. For existing applications, this application service object is served by a separate application service process that converts the accessor requests into instructions for the application. Newly developed applications can directly accept requests to the corresponding application service object. A sequence of accessor requests can be stored as a named program and processed by the accessor server upon request. The accessor object-base can be edited so as to contain the most recent information about the available data and function. The accessor framework has been made operational in a network with servers and workstations.

keywords: integration, workstation, distributed processing, object-orientation

1 Introduction

The advent of networked, graphical workstations has brought a host of data, knowledge, and software within the reach of end-users. In spite of this development, the potential of network technology has largely been limited to individual, closed, and stand-alone existing applications together with user-controlled network file transfer. A review of this lack of software integration was given by Meyers [1]. Opposite to this closed attitude of software is the open systems concept [2]. This concept ideally covers the following issues:

1. Exchange of data among applications without a need for user intervention,
2. A view on applications that regards them as repositories of data and functions that can be utilized by other applications,
3. Contribution of newly developed applications to the set of available functions in the network computing environment,
4. Inter-connection of applications through the network, exchanging request messages from a client application to a server application.

To accomplish these goals we developed an architecture for software integration and inter-application communication. The architecture has been designed according to the open systems concept to allow for the incorporation of existing applications virtually without the necessity of any modification. In our definition, integration is the process of inter-connecting autonomous software applications within a network, enabling the automatic transfer of data, functions and knowledge without human intervention.

1.1 Application domain

Applications of computers - especially those in medicine - are characterized by a wide variety of different software packages, covering direct patient care, health administration, and research. Clinical users are increasingly confronted with the fact that their data are distributed among various databases. Furthermore, a wide range of systems for statistical analysis, graphical presentation, signal or image analysis, and interpretation or decision support is available on mainframes, PCs, and workstations. In order to take full advantage of these existing applications, a user-friendly environment is required that integrates existing databases on central computers with PC or workstation applications and that provides a mechanism to invoke data processing on all computers in the network. This requires inter-application communication and should give users processing support and reduce their involvement in managing data and accessing individual databases and applications. In addition, it should allow them to use functions within an application without them having to know what application must be called.

In general, existing software applications are not suited for inter-process communication. Differences between the views of vendors and software developers on the data representation and the command interface, and the inability to communicate with the application through a network restrict the use of existing applications in an open system's environment. Modifying applications to adhere to a common interchange format is laborious for large systems, such as hospital information systems, and increases the maintenance. Furthermore, modifications to consolidated software, such as statistical applications, endangers its reliability.

The ultimate goal of our software integration project is to give the end-user, who typically is no software specialist, support that better suits his requirements. In this integrated environment, the data flow between the applications is fully automatic, and a common applications command interface and a user-interface provide the user with a simple and intuitive view on data and functions. In a prototype, an integration architecture was developed, and this integration concept was tested for its feasibility [3]. The potential of this integrated prototype has also been assessed in a user evaluation.

This integration architecture also offers a software platform for newly developed applications that follow the open systems concept: independent modules or services in the network are designed as to contain functions that may be shared by different applications. For passing requests from a client to a service, a special accessor language was defined. Each service takes care of a specific set of requests, provided by the application, and responds with an answer. Together, the services collaborate to provide the total functionality of an application.

1.2 Accessor framework

For our software integration project, an Accessor Framework was developed that controls the delegation of requests to the various application services (see Figure 1). This Accessor Framework is necessary to make a request from a client, independent of the service that resolves it: the accessor object representing the request is linked with an application service object. This Accessor Framework provides the following support:

1. It binds a request to a particular application service at run-time, thus allowing flexible incorporation of new services and new functions.
2. Since an application service object is associated with an application service process, the Accessor Framework translates the request into the proper invocation of the function in the service, and reformats data provided with the request into the application data format. This application service process returns results in the Accessor Format.
3. The Accessor Framework concentrates the application dependencies in one overall model and explicitly specifies the linking of a request to a service accessor and of the service accessor to the application service and finally to the application.

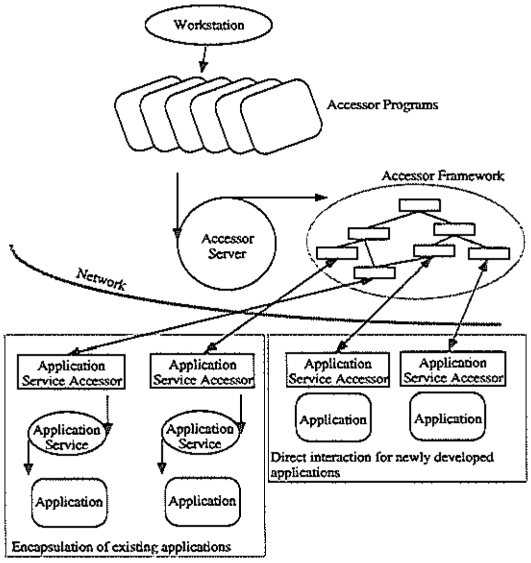

Workstation
Accessor Programs
Accessor Framework
Accessor Server
Network
Application Service Accessor
Application Service Accessor
Application Service Accessor
Application Service Accessor
Application Service
Application Service
Application
Application
Application
Application
Direct interaction for newly developed applications
Encapsulation of existing applications

Figure 1. Overview of the accessor environment. From a workstation, accessor programs can be activated and passed to an Accessor Server, that will search the Accessor Framework for an accessor object that corresponds with the accessor request. A method associated with the accessor object will either forward the request to other accessor objects, or will directly forward the request to an Application Service Accessor. These Applications Service Accessors are located on various hosts and hide the interaction details with an application. Applications can be either existing applications embedded in an Application Service or newly developed applications that are suited to directly interact with the Application Service Accessor.

The Accessor Framework functions as an additional integration layer. Its flexibility is a prerequisite for a rapidly changing application domain, such as medicine; changing database definitions, new application releases, and new developments demand an environment that can easily and dynamically cope with latest developments. The Accessor Framework can be graphically edited to reflect the computational environment.

1.2.1 Five integration layers

In our software integration project, five levels of integration have been described [4]: integration of hardware, data, functions, user-interfaces, and nomenclature. These five levels define communication at OSI's application communication layer [5]. These integration levels should be covered by the Accessor Framework. The *hardware integration* level connects different computers with different operating system on the application level. Requests are routed by the Accessor Framework from the client process through the network to the server. The requests contain specifications of data and/or functions that are supported by the applications. The requests are expressed in an Accessor Framework Language that defines a common data-interchange format. Via this data-interchange format, applications exchange data, thus supporting *data integration.* The Accessor Framework Language is augmented with an Accessor Framework Command Language that is used to express function requests together with a uniform parameter-passing mechanism. *Commands* of this common Language are translated into application-specific instructions. The Accessor Framework also supports two *user-interface integration* paths: (1) communication at X11 level and (2) through a set of user-interface function requests that can be passed to an X11 user-interface server. The Accessor Framework provides, finally, *integration at the semantic level* by definition of a nomenclature that contains a common data dictionary and a vocabulary for coding. The data dictionary and vocabulary are managed by the Accessor Framework.

1.3 Related research

Software integration research is still at its very beginning. This makes it difficult to have all the different research projects described in the literature categorized along a number of comparable features. Consequently, some integration research projects are reported here just to show the different approaches to integration. To relate our present research project with those of other investigators, we summarize closely related projects.

Distributed object-based programming systems have been a research topic for some years [6]. This research has been directed at environments that allow the development of applications consisting of a set of interacting modules. These different modules are usually assigned to different processors and provide a natural concurrent programming mechanism. Examples of these systems are Tanenbaum's Amoeba project [7] and the Chorus system of Banino [8]. The remote procedure call (RPC) mechanism has been investigated by others to support open distributed processing [9].

1.3.1 "Tight" and "Loose" integration

Some approaches stress the interconnection of applications in a network and focus less on the programming language. This approach can be divided into two main streams: (1) the definition of a common integration layer between the applications, i.e., "loose" integration, and (2) connecting the applications directly, i.e., "tight" integration. Though a common integration layer allows extension with other applications, its integration bandwidth is limited, because it cannot use all properties of the applications. The latter approach maximizes the inter-operability of the applications but requires a stable, not changing environment. An example of the "tight" integration approach is multi-database research as reported by Landers et al. [10] and Bertino et al. [11]. The limited domain of database management systems enables a very direct connection without additional integration layers. Other examples are the integration of CAD-CAM systems [12] and of databases with knowledge bases [13].

Barsalou and Wiederhold [14] explored the use of mediators as a mechanism to abstract from the access to data. Their Penguin system combines an expert system with a database management system via these mediators. Although this example of "loose" integration is restricted to data, the mediator concept proves to be powerful and accessible for many application views. A similar abstraction has been designed by Greenes who proposes a "daxel", data access element [15]. These daxels support an environment that permits applications to access data without having to know the details of where and how data are stored.

Hewlett Packard's Physician's Workstation is an attempt to couple existing database applications in a general client-server architecture [16]. A similar strategy has been followed by the Field environment that integrates software development tools as services [17]. Such a connection is limited, however, to the exchange of information about the status of the various tools. The Helios software bus architecture is one of the research projects that focuses on new applications/services that can be inter-connected at the data level in a client-server model [18]. This research is primarily directed at an object-based connection mechanism.

Large-scale integration projects have been started to integrate a broad range of databases with applications. The Integrated Academic Information Management System (IAIMS) project of the National Library of Medicine aims at connecting various information systems so that clinicians can access them from any workstation in the hospital [19]. Primary attention has been paid to the realization of a network topology and protocol that allows all hardware to communicate.

In the remainder of this paper, the requirements for this integration architecture will be defined, and the elements of the Accessor Framework are described. Various classes of accessors will be presented together with a language that can be used to address the accessors and an accessor server that manages the accessor objects.

2 Further requirements

Experience with the development of a prototype of a software integration environment [3,4] taught us that the following elements should be covered in our integration architecture:

1. *Encapsulation*. Existing applications should be embedded in an application service. This service translates common requests to the specific formats used by the application. Typically, a data translation facility transforms data between the common interchange format and the application format. A command generator expands a function request into a series of keystrokes or commands for the application. Communication at the user-interface is either arranged via the X11 communication protocol or via a X11 user-interface service. Each application has its own specific application service.

2. *Shareability.* Applications should be shared by various clients at the same time. This requires application services to activate applications more than once at the same time. The application service should always be ready to accept requests.
3. *Connectivity.* The connection between a client and a server should be established dynamically. A *router* should consult a database to find a service somewhere in the network that can handle the request. New applications can be added by editing the database and completion with the information of the new application service. Clients never directly address an application service, but should always utilize the Accessor Framework to establish a connection, in order to take advantage of the most recent developments. This database also contains information about additional parameters and defaults a request should be completed with.
4. *User Configurability.* The Accessor Framework allows users to specify for each type of request the preferred application service and for each service the preferred host that contains the service. Changing or adding request parameters can be specified by each user independently.
5. *User Friendliness.* Although this is outside the scope of the integration architecture, attention should be paid to how the functionality is presented. The Accessor Framework allows different user-interface services for each of the user groups. Depending on tasks and experience, a user can select a particular type of user interface that best fits his view on the computing environment.

3 Accessor framework model

The Accessor Framework consists of (1) an Accessor object-base that contains information about the access to data and functions, (2) operators that handle requests for objects, and (3) various types of accessors for the different elements available in the environment: hosts, applications, parameters, requests (data, functions, presentation forms, and knowledge). In addition, (4) an Accessor Language is described that can be used to uniformly specify requests to all accessor objects, and (5) an accessor server that accepts these requests, searches the object-base for (6) an application service that accepts the requests, and forwards the request to an application service that translates the request to the application-specific instructions.

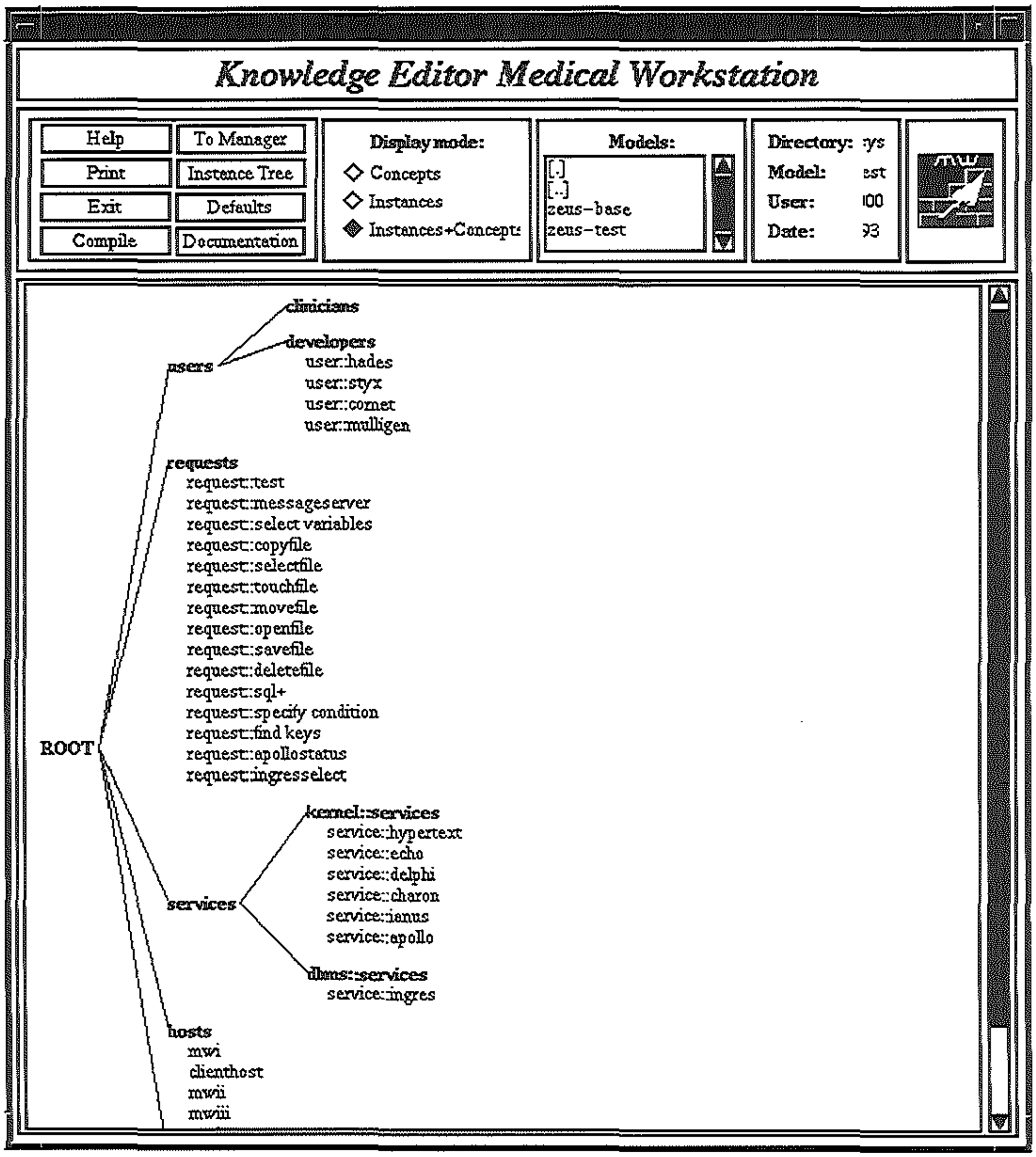
3.1 The accessor object base

The Accessor Framework is organized as an object-based environment, an object being an entity that encapsulates information and associated operators. The information and the implementation of the operators is in general completely protected, and hidden from other objects and the outside world. Information can be obtained by invoking a special operator, associated with the object. According to the object-oriented paradigm, these operators create a well-defined interface that hides the object's information and the operator specification [20].

Objects are grouped in *classes*. A class also defines a template from which objects can be created. Each object is an instance of one class only. A group of objects that possess the same information and the same set of operators belong to the same class. Information fields of an object are called *aspects*. New classes can also be defined by specifying how the new classes differ from the original class. Aspects specified in the original class are then *inherited* in the new classes. The new class is then a subclass of the original base class. The relation between the classes is sometimes referred to as an "is-a" or a "is-a-kind-of" relation. The Accessor Framework permits classes to have one base class only, so multiple inheritance is not supported. Operations applied to a class are applied to all instances in that class and to all instances of each subclass.

Objects can be linked to each other to represent certain relationships. Typical relationships represented by these links in the Accessor Framework are: "is_provided_by", "is_resident_on", and "consists_of_parameters". Each object in the Accessor Framework corresponds with requests, services, hosts, and parameters. A request object is linked with the services that can handle the request. Each service is in its turn linked with the hosts that provide that service. Each request object is also linked with the parameter objects representing the parameters that can be provided with the request.

In Figure 2, an Accessor Framework is shown. The bold terms represent the classes or concepts available in the Accessor Framework. The terms below a class represent the instances that can be found in the Accessor Framework. The lines between the classes indicate the "is-a" or "is-a-kind-of" relation between classes.



Display mode: ◇ Concepts ◇ Instances [✓ Instances+Concepts]

```
ROOT
  users
    clinicians
    developers
      user::hades
      user::styx
      user::comet
      user::mulligen
  requests
    request::test
    request::messageserver
    request::select variables
    request::copyfile
    request::selectfile
    request::touchfile
    request::movefile
    request::openfile
    request::savefile
    request::deletefile
    request::sql+
    request::specify condition
    request::find keys
    request::apollostatus
    request::ingresselect
  services
    kernel::services
      service::hypertext
      service::echo
      service::delphi
      service::charon
      service::janus
      service::apollo
    dbms::services
      service::ingres
  hosts
    mwi
    clienthost
    mwii
    mwiii
```

Figure 2. The object-oriented Accessor Framework editor with an example Framework. Bold terms represent concepts, the terms below concepts represent instances and the lines between concepts represent "is-a" relations.

All accessor objects have an aspect containing the names of the services that can respond to the request, i.e., an aspect for the pretty name of the accessor; for the name of the class it belongs to; for the creation, the last access and the modification time; for the author's name; for access-control information; and for storing the operations allowed. In addition, each accessor class can add its own specific aspects.

3.2 Operators

When addressing a request to an object, an operator is needed to handle the request. The Accessor Framework allows three levels that can contain operators. First, each object can have a list of associated user-defined operators. The accessor server will search the list of associated operators for the requested operator. Operations that are identical for all objects in a class can be specified at class level by the user. If an operation is not available at object level, the class and subsequently its superclasses are searched for the specified operator. If the operator is not specified at object and class level, the built-in operators of the accessor server itself are employed.

A set of built-in operators is always accessible to all objects. These operators include a print operator for printing the contents of an object, an edit operator for changing the contents, a create and destroy object operator, a create and destroy class operator, a set and get aspect operator to edit the values of object aspects, and a get_subclasses, get_superclass and get_objects operator. In general, these built-in operators are powerful enough to resolve the request. However, sometimes it is necessary to provide alternative operators that provide an alternative resolution algorithm. For instance, if the resolution algorithm uses distance as a measure to select an application, an alternative resolution algorithm could try to optimize for response time.

All accessors and operators can be accessed through an accessor language that is also described in this paper.

3.3 Accessors

3.3.1 Host accessor

The host accessor contains per accessible computer in the network an accessor object. This accessor object contains aspects that specify the network protocol used by the accessor server for communication. Two operators are associated with each host accessor, a send and a receive operation to transport requests between the Accessor Framework and an application service on that host. The built-in send and receive operator assumes a TCP/IP Ethernet connection. The host accessor can also contain abstraction of emulated hardware. The send operation to services that run on emulated hardware initiates the emulation and opens a communication channel using this emulation process.

3.3.2 Application service accessor

Application service accessors hide the access to the services available in the network. A service accessor contains operations to compose a request to the service that is present and to decompose a received request to the standard request format of the Accessor Framework. User-defined compose and decompose operations enable incorporation of services in the workstation that have a deviating request syntax. The built-in compose and decompose operations assume application services to accept common Accessor Framework requests. The service accessor contains a reference to a host accessor that can transport the request to the application service.

3.3.3 Parameter accessor

This class of accessors is used to specify the parameters that can accompany a request. Each parameter accessor contains an aspect specifying whether the parameter is mandatory or optional, an aspect that contains the default value for the parameter, and an aspect that contains the valid values for this parameter.

An "obtain" operation can be attached to each parameter accessor to query the user about the actual parameter value. This operation is only executed when the value is required and not specified.

3.3.4 Request accessor

A request accessor corresponds to a request that can be sent to the Accessor Framework. The name of the request is identical to the name of the corresponding request accessor. The request accessors are divided into the following groups: data accessors, function accessors, presentation accessors, and knowledge accessors. A request accessor is always linked with a service accessor. Depending on the request, the accessor server determines which service should be used to resolve the request.

3.3.4.1 Data accessor

A data accessor corresponds to a data attribute in one of the databases. The data accessor contains information to access the attributes in the various heterogeneous databases and files that correspond with the data attribute represented by the accessor. In addition to the information necessary for obtaining the data, the data accessor also contains dictionary information such as the minimum and maximum values, the valid codes, the type and size of the data, etc. Data accessors are grouped in classes according to a data model. This data model allows users to select classes of data rather than data of the individual data accessors.

For each attribute in a database that corresponds with the medical datum represented by the accessor, an address is stored. This address specifies a tuple <database management system, database, relation, attribute> that identifies the data and a list of the keys that should be selected when joining the data. When the address is not specified in the request and the same data are available in multiple databases, the user is requested to indicate what database should be used. Selection of data at class level will result in individual requests for the objects in that class (or subclasses), followed by a join operation of the individual results.

The accessor server determines from the database management system part in the attribute address parameter of the request, which service linked with the data accessor should provide the data. The request is then forwarded to the specific service accessor.

Each request for data can be completed with a selection condition. A condition specifies criteria the data should fulfil. In case of a conditional selection, the condition-evaluation service is called with both data and the selection condition(s), or with a uniqueness constraint. The condition-evaluation service filters the records according to earlier defined criteria, and the uniqueness constraint specifies how records with identical keys have to be removed. The condition-evaluation service responds with data that fulfil the condition and uniqueness criteria. Subsequently, the data are returned to the client that invoked a request to the Accessor Framework.

Data accessors have the following additional aspects containing:

1. the *type* of the data, i.e., numerical, textual, coded,
2. the *format* of the data, i.e., the size and the structure,
3. the *corresponding codes* in the various classification systems necessary for matching coded data from different databases with different classification systems,
4. a list of the *valid codes* for the corresponding data attribute,
5. a list of *addresses and keys*; these addresses can be used to locate the data attribute and the keys to uniquely address the records,
6. the *lower and upper bound* for numerical data,
7. *triggers*; the names of function accessors that have to be accessed when retrieving or inserting data. These function accessors test the data for fulfilling particular criteria,
8. *uniqueness*; this aspect specifies whether records can occur with the same keys. A maximum, minimum, average, or time criterium can be used to select a particular record from the set of records with the same keys,
9. a list of *service accessors* corresponding with the database management systems specified in the addresses,
10. *synonyms* for the name of the data accessor; references to data included in the synonyms list are also mapped to this data accessor.

Operations associated with a data accessor are a select, insert, and update operation. These operations compose an SQL query that can be forwarded to the service. The service will subsequently translate these SQL queries into the proper instructions for the database management system and reformat the data in the proper format. Retrieval results are reformatted to the common data interchange format and returned to the Accessor Framework.

Temporary data accessors can also be created. These temporary data accessors facilitate the abstraction of files that have been created as a computational result. The address aspect of the data accessor contains for these accessors the name of the file. These temporary data accessors are put into a specific data accessor class according to the application by which they are generated. A set of operations allows applications to extract data from these files or to translate them to the standard data storage format. This set of operations depends on the type of temporary data that should be accessed.

3.3.4.2 Function accessor

The applications accessible from a workstation contain a wide range of data processing functions. Through function accessors, each of these functions can be accessed as an individual function and the corresponding application is automatically started. Similar to the data accessors, function accessors are grouped into classes according to the type of functions they represent. The function accessor contains a list of application services that may respond to the request.

The function accessors contain a link to a list of parameters that are required for the function. A complete request is forwarded to an application service accessor that is linked with the function accessor. The service accessor will then forward the request to the application service, translate any results to the common data interchange format, and create temporary data accessors to control the access to that result file.

The application service will accept the request from the service accessor and correspondingly generate a script of commands or keystrokes. Subsequently, the application is started and the script or the keystrokes are entered into the application. The results are captured, translated into the common data interchange format, and redirected to the Accessor Framework, which in turn, returns the results to the client that invoked the request.

The Function accessors aspects that are defined are the following:

1. *synonyms* allowing functions to be referred to by other names,
2. *password required*, specifying whether a request should contain a password,
3. *application name*, containing the name of the application that provides the function,
4 *version/release*, specifying the version number or the release date of the application.

There are two types of function accessors: for presentation and for knowledge incorporation.

3.3.4.2.1 Function accessor for presentation

Presentation accessors are special function accessors; in contrast to function accessors, presentation accessors do not return any result other than visible on the screen. Presentation accessors are grouped into classes according to the input format they accept: lists, nxn tables, 3D tables, textual information, signals, images, etc.

The presentation accessor is linked with presentation services, each of them providing a set of presentation functions and residing on a particular host. The presentation services create an X11 interface that displays data via the network on the user's workstation.

The presentation accessors provide a uniform access to graphical windows on the screen. A request for a presentation form contains the data in the common data interchange format. Each presentation accessor specifies whether data updates should be passed to the same presentation form, or should be handled by a new presentation form.

For particular functions, it is necessary to set up a stream mode. Presentation accessors that run in stream mode, continuously accept information and update their presentation with the new information. Typically, this mode is used for a monitoring function (e.g., for signals) or when the data used in the presentation form is not available in one request. The latter situation occurs when the selected dataset is too large to be stored on the user's workstation or to be contained in one request.

The presentation accessors also provide virtual terminals on various hosts. Through these terminals, the user can interact with the operating system, or start specific functions.

Although currently the presentation accessors provide access to functions that obey the OSF/Motif style guide and run on the X11 protocol, new developments could also utilize other user-interface management systems. For platforms following other graphical user-interface systems, new presentation services can be defined, and the selection of the presentation service is then influenced by the available system. Additional graphical user-interface systems include Windows 3.0, Sun's OpenLook, and Macintosh's tool-box.

Presentation accessors contain the following aspects:
1. *synonyms*, allowing requests with alternative names to be mapped to this accessor,
2. *user-interface management system*, specifying what window system is used for presentation,
3. *input format*, defining what type of input is expected (e.g., table, array),
4. *stream mode*, specifying whether this accessor can be run in stream mode.

3.3.4.2.2 Function accessor for knowledge

Knowledge accessors abstract the access to expert decision rules that can be used, e.g., for assessing the data. These knowledge accessors provide access to a range of services that encapsulate expert systems and inference engines. Knowledge accessors compose an evoke instruction for an application service. This application service will subsequently translate the evoke request to commands that activate the inference engine with the appropriate rules or frames. The knowledge accessors abstract the client from the implementation of the expert system and the knowledge representation.

The knowledge accessors can be grouped in classes according to the type of problems they deal with. For some classes, multiple knowledge accessors can be defined. This allows the consultation of more than one rule or frame for the specific problem.

Knowledge accessors can be evoked indirectly as triggers by data accessors to test particular conditions of the data. Each knowledge accessor can also be directly evoked by a client. An attribute aspect specifies what data should accompany the request to the service. When the knowledge accessor is invoked as a trigger, the data are automatically passed in the request. If the knowledge request is directly specified by a client, the knowledge accessor will send a select request to the data accessors specified.

The knowledge accessor contains the following aspects (additional to the standard accessor apects):

1. *synonyms*, specifying a list of names that can be used alternatively for referring to this knowledge accessor,
2. *data required*, containing a list of data accessor names that should be accessed before a knowledge accessor can be activated,
3. *name of expert system*, specifying the name of the expert system application or of the knowledge base that contains the knowledge,
4. *release/version*, containing the release and/or the version of the expert system application.

3.4 Accessor language

The accessor language specifies the conventions that are used within the Acessor Framework. It consists of the common data interchange format, and the command and parameter mechanism. Requests can be directed to the accessor server.

Each parameter specified in a request overrules the parameters defined in the Accessor Framework. This allows users to tailor the behavior of the system according to their own preferences. The accessor language contains data in an Intermediate Storage Format (ISF). This ISF has been defined for the prototype software integration system and contains data in ASCII, together with a format specification. This format can be a string specifying the type and the size of the data records, or it can contain a field separator. The format also contains the names of the data accessors for each of the variables.

Requests can be simply specified by the accessor language. Each request begins with the keyword *begin* and ends with the keyword *end*. The request is specified as a list of parameters with a value. The following parameters are mandatory for a request:

1. *message*. This parameter is followed by the name of the request,
2. *user*. This contains the name of the user that issued the request,
3. *display*. This specifies from what display the user is working. All interactive responses are directed to that display,
4. *from*. This parameter specifies the name of the client that composed the request,
5. *message id*. This is a unique number identifying the request. All results directed towards this request use the same number.

In addition to these mandatory fields, the client can specify parameters that overrule the parameters available from the Accessor Framework. An example of a request is outlined in Figure 3.

Accessor Request

```
begin
 message:select data
 user:Van Mulligen
 display:mwi:0.0
 from:sqlselect
 message id:sqlselect#001
 query:select patientnr, sex, name from his.regist.andro where sex = "M"
end

begin
 message:save data
 user:Van Mulligen
 display:mwi:0.0
 from:sqlselect
 message id:sqlselect#002
 host:ote.mi.fgg.eur.nl
 filename:"first_selection"
 expiration: 12-08-92
end
```

Figure 3. An example Accessor Request. This example specifies a request to select data from any database that contains the data, followed by a request to save the results in a file on a particular host in the network.

underline:	required fields for a message, provided by the application,
italic:	required fields provided by the user or the Accessor Framework,
underline&italic:	optional fields, provided by the user,
bold:	message keywords.

Sequences of requests can be stored as workstation programs. These workstation programs can be sent to the accessor server and provide the user with typical tasks: e.g. daily overviews, such as reports of the patients that were treated, trend analysis, risk analysis, graphical presentations, etc. Each user can record his actions during a session as a new workstation program. This new program can be activated and performs as a high-level task for the user.

3.5 Accessor server

The accessor server is the process that accepts invoking requests and passes them on to the application service. The accessor server interacts with its accessor base to complete the request, find an application service that best fits the request, and forwards the request to the application service. If the accessor server cannot resolve the request, it will forward the request to a next accessor on a different host. If this accessor can resolve the request, it will return to the initial accessor server a completed request, together with the name of the service that can accept the request. The accessor server will consecutively interrogate all known accessor servers in the network until one responds with a resolution. If no server responds with a resolution, the request cannot be solved and the initial accessor server will inform the user about this.

3.6 Application service

Each application service encapsulates an existing application. The application service consists in general of three drivers: a data-translation driver that translates data from the ISF to the application specific format, a driver that translates results to the ISF, and a driver that expands commands to the application-specific command or keystroke sequences.

4 Evaluation

One of the main reasons for developing the accessor framework is to reduce the work involved in maintaining an integrated software environment that is based on various applications. The programming effort to develop software links between applications for exchange of data and commands can be reduced significantly through a common data interchange and command language (as the accessor framework is). When developing links between applications, the number of interfaces to be developed for N applications is N(N-1). The accessor framework reduces this number to N interfaces. When the number of applications is large, the additional effort introduced by the accessor framework is easily compensated by the gain in developing interfaces.

The overhead introduced by the accessor framework has been tested in the prototype on a Hewlett Packard HP9000/385 computer. The accessor framework did not add to the delay caused by the network communication between the applications. For the prototype, however, the accessor framework was not fully completed nor did the programming environment contain all final features.

The prototype integrated medical workstation has been developed in which support of clinical research was provided by integrating various databases with statistical and graphical presentation applications. Although the prototype was not developed with the accessor framework as described in this article, it is from a functional/user perspective equivalent to the accessor framework integration environment. The evaluation of this prototype was accomplished as follows.

After an introductory course, 26 users (11 biomedical researchers and 15 clinicians) attempted to solve 6 representative clinical research problems with the integrated medical workstation. During the experiment, all actions were logged and timed and all intermediate and final results were stored. After the experiment, users were also asked to complete a form to express their experience with the integrated workstation. For each question, its status was determined, being one of: correctly completed, incorrectly completed, not completed and not done. In Table 1, the results are shown for all users. Note that a number of users did skip questions not because they could not answer them, but because of other reasons (time, repetition etc.).

Table 1. Results from the user evaluation. Per question, the number of correctly completed, incorrectly completed, not completed and not performed questions is shown for all users.

Questions for clinicians and biomedical researchers (N=26)

Question	Completed&Correct	Completed&Incorrect	Not Completed	Not Done
1	22	4	0	1
2A	19	4	4	1
2B	17	4	0	6
3	23	0	1	3
4A	22	0	0	5
4B	19	1	0	7
5A	17	5	0	5
5B	13	0	0	14
6	19	0	0	8
Total	**171**	**18**	**5**	**50**

Footnote Table 1: A number of questions exists of two parts (A and B). These are shown separately, and one can clearly see that B parts are often skipped.

The biomedical researchers were not convinced about the time savings resulting from using the system, but they found the workstation approach on the average better than their current environment, which is in general a collection of stand-alone applications. Clinicians were clearly more positive about the system and judged the time savings significant and liked the system better than their current support. Novice users clearly appreciated the integration more than experienced users, such as biomedical researchers (see Tables 2 and 3).

Table 2. Results from the user evaluation. Subjectively estimated time savings for biomedical researchers and clinicians as compared to current support.

Time savings	Investigators	Clinicians	Total
Poor	3	0	3
Reasonable	5	7	12
Good	3	8	11
Total	11	15	26

Table 3. Results from the user evaluation. Rating of subjective appreciation by biomedical researchers and clinicians of the workstation, as compared to current support.

Comparison	Investigators	Clinicians	Total
Poor	1	0	1
Reasonable	7	1	8
Good	3	14	17
Total	11	15	26

5 Discussion

The Accessor Framework as described in this paper offers a general integration architecture that can be used to dynamically connect clients with services in a network. The object-base containing the access details of all available data and functions can be consulted by the Accessor Framework to select an application service that can handle the request.

Although experience with integration in our prototype taught us that the client-server model is most appropriate for inter-connecting applications, a complete client-server model has only recently been implemented. Attention should be paid to performance aspects of this architecture. All requests are efficiently handled by the accessor server, but the dynamical binding of a request to an application service also requires overhead. Concurrent accessor servers can, however, be implemented on various hosts to improve the performance and allow applications to connect to the closest available application service rather than possibly starting a long-distance connection. In principle, therefore, the limiting factor is the communication network and not the servers. The Accessor Framework allows for multiple, and even parallel servers in the network. When independent, requests can be forwarded by the Accessor Framework without waiting for a reply, thus making parallel actions possible.

Experience with redundant resources is still limited. Currently, the user has to specify which application service to use in case of multiple choices. Research should be directed at models that allow a more advanced algorithm in resolving redundancy, perhaps using context information to guide the selection of services.

Although the Accessor Framework provides a flexible integration architecture for existing applications, the construction of application services remains a research issue. Current research focuses on the development of tools that allow easy construction of the necessary drivers.

This Accessor Framework is a successful proposal that underwent evaluation in a prototype, to define an inter-process communication protocol. Currently, an architectural approach is used to standardize the Open Distributed Processing protocols [21]. This international standardization effort defines a unifying framework that ensures interoperability between existing and future computational resources. It is still to be investigated how these standards can also be incorporated in the Accessor Framework.

Acknowledgement
The authors gratefully acknowledge the comments of professor Gio Wiederhold, Stanford University, California, USA. Part of this work has benefitted from a grant of Hewlett-Packard Medical Products Group Europe.

References

[1] MEYERS, S. Difficulties in integrating multiview development systems. IEEE Softw. *3* (1991), 49-57.

[2] KUO, J.H. Open architecture and tool integration of software development environments. In *Proceedings of the First Conference on Systems Integration '91*, (Kyoto, April,1991). IEEE Computer Society Press, Los Alamitos, Ca., 1991, pp. 444-453.

[3] VAN MULLIGEN, E.M., TIMMERS, T., LEAO, B. DE F. Implementation of a medical workstation for research support in cardiology. In *Proceedings of the 14th Symposium on Computer Applications in Medical Care* (Washington DC, November, 1990). IEEE Computer Society Press, New York, 1990, pp. 769-773.

[4] VAN MULLIGEN, E.M., TIMMERS, T., LANGHOUT, A., AND LEAO, B. DE F. CW2000 - a medical workstation to support research in cardiology. In *Proceedings Computers in Cardiology 1990* (Chicago, September, 1990). IEEE Computer Society Press, Los Alamitos, Ca., pp. 203-206.

[5] ZIMMERMANN, H. OSI reference model - the ISO model of architecture for open systems interconnection. *IEEE Trans. Commun. 28* (1980), pp. 425-432.

[6] CHIN, R.S., AND CHANSON, S.T. Distributed object-based programming systems. *ACM Computing Surveys 23, 1* (1991), pp. 91-122.

[7] TANENBAUM, A.S., AND VAN RENESSE, R. Distributed operating systems. *ACM Computing Surveys 17, 4* (1985), pp. 419-70

[8] ROZIER, M., AND MARTINS, J.L. The chorus distributed operating system: some design issues. In *Distributed Operating Systems. Theory and Practice.* Springer-Verlag, Berlin, Heidelberg. pp. 262-287

[9] ROBINSON, D. Remote procedure call: a stepping stone towards ODP. *Computer Networks and ISDN Systems 23* (1991), pp. 191-194.

[10] LANDERS, T., AND ROSENBERG, R.L. An overview of multibase. In *Distributed Data Bases*, H.J. Schneider ed., North-Holland, Amsterdam (1982).

[11] BERTINO, E., NEGRI, M., PELAGATTI, G., AND SBATELLA, L. Integration of heterogeneous database applications through an object-oriented interface. *Information Systems, 14, 5* (1989), pp. 407-420.

[12] HSU, C., AND SKEVINGTON, C. Integration of data and knowledge in manufacturing enterprises: a conceptual framework. *Journal of Manufacturing Systems 6, 4* (1986), 277-285.

[13] PEDERSON, C.H., ANNEVELINK, J., BARMAN, H., DERRETT, N., SEABORNE, A., SYRETT, M., AND TOFT, P. Data and knowledge bases as integral parts of a distributed object infrastructure. In *Proceedings of the First International Workshop on Interoperability in Multidatabase Systems '91* (Kyoto, April, 1991), IEEE Computer Society Press, Los Alamitos, Ca., pp. 101-107.

[14] BARSALOU, T., AND WIEDERHOLD, G. Knowledge-directed mediation between application objects and base data. In *Proceedings of the Working Conference on Data and Knowledge Base Integration* (October, 1989).

[15] GREENES, R.A. Promoting productivity by propagating the practice of "plug-compatible" programming. In *Proceedings of the fourteenth Symposium on Computer Applications in Medical Care* (Washington DC, November, 1990), IEEE Computer Society Press, New York, pp. 22-26.

[16] TANG, M., ANNEVELINK, J., FAFCHAMPS, D., STANTON, W.M., AND YOUNG, C. Physicians' workstations: integrated information management for clinicians. In *Proceedings of the fifteenth Annual Symposium on Computer Applications in Medical Care* (Washington DC, November, 1991), IEEE Computer Society Press, New York, pp. 569-573.

[17] REISS, S.P. Connecting tools using message passing in the field environment. *IEEE Software 4* (1990), pp. 57-66.

[18] DEGOULET, P., COIGNARD, F.C.J., LAURENT, M.C., LUCAS, L., BEN SAID, M., MEINZER, H.P., ENGELMANN, U., SPRINGUB, A., BAUD, R., AND SCHERRER, J-R. The HELIOS european project on software engineering in medical informatics. In *Proceedings of the IMIA Working Conference on Software Engineering in Medical Informatics* (Amsterdam, October, 1990). North-Holland, Amsterdam, pp. 125-137.

[19] LUNIN, L.F., AND BALL, M.J. Perspectives on Integrated Information Management Systems (IAIMS). *Journal of the American Society for Information Science. 39, 3* (1988), 102-112.

[20] STEFIK, M., AND BOBROW, D.G. Object-oriented programming: themes and variations. *The AI Magazine* (1985). 40-45.

[21] BOWEN, D. Open distributed processing. In *Computer Network and ISDN Systems 23* (1991). Elsevier Science Publishers, pp. 195-201.

CHAPTER 6

New Perspectives On An Integrated Medical Workstation

E.M. van Mulligen[1,2], T. Timmers[1], J.H. van Bemmel[1]

[1]Department of Medical Informatics, Erasmus University Rotterdam, The Netherlands
[2]University Hospital Dijkzigt Rotterdam, The Netherlands

(to be published)

Abstract

Medical workstations are increasingly introduced as the integration center of various information streams. This paper describes the intrinsic features for integration as proposed by Greenes [1] and others, connectivity, modularization, shareability, abstraction and encapsulation, and their impact on an integration architecture for a medical workstation that integrates existing applications. Using these features as an onset, a new vision on a medical workstation will be outlined. The description of these features and the deduced requirements are followed by an example of a new architecture for such an integrated medical workstation.

keywords: workstation, integration, encapsulation, abstraction

1 Introduction

Medical knowledge and information is growing so rapidly that clinicians, medical researchers, and students of medicine are increasingly beset by the problems of data management and the organization of the vast amounts of data collected during patient encounters and by various medical devices. Paradoxically, this data overload causes clinical information underload for most clinicians.
Attempts of medical computer scientists to adopt information technology to alleviate these managerial and organizational problems of medical professionals lack a fundamental coherent approach to the key issues of information integration. Use of existing resources (data, knowledge, and software), dispersion of data through a network environment, heterogeneity of resources, distributed processing, graphical user-interfacing, and dynamic extensibility are rather treated as separate problems rather than as a coherent issue. Although the technologies of networks and graphical user-interfaces have matured to an extent where standards have evolved, research addressing integration technology has received much attention only recently [2]. In spite of the diversity of software and the predominance of the traditional paradigm of stand-alone, isolated software tools, impeding integration of existing applications, the undervalued profits of this perspective are immense: a host of data and functionality, no reinvention of the wheel, no repetition of software development errors of the past, and truly task-oriented support for the clinician.

Our initial experience with a prototype medical workstation and related research have constituted the recognition of the intrinsic features necessary for this paradigm shift to integrated, open software environments, and consequently, for an integrated medical workstation. In the following, these features are described with their derived functional requirements, and a new architectural vision satisfying these features is discussed. Although these features do not necessarily imply task-oriented support, they can be regarded as the essential ingredients for an integrated workstation for the clinician (Figure 1).

2 Prototype medical workstation

In spite of the presence of large-volume databases, extensive reliable software, powerful graphical workstations, and network connections, most clinicians are unable to satisfactorily solve their clinical research tasks with this environment. The skills required to

objective	covered in the architecture by
	User Interface
uniformity	Accessor scripts
	intermediate storage format
integration	ARPA client
	-------- network communication
	ARPA service
sharing	
	Encapsulation software
	command facility
modularization	data translation facility
	Application

Figure 1. The relation between the architectural components and the integration features.

extract data, to address the functions in miscellaneous software packages, and to exchange data among various hardware platforms, surpasses the capabilities of most clinicians. The aim of the prototype medical workstation is to offer a user-friendly graphical environment that connects existing databases and data processing functions in a network, liberating the user from access and management details [3].

In this prototype, existing applications were encapsulated to realize the intrinsic integration features discussed above. The integration architecture that supports clinical research has been obtained by providing each application with an individual encapsulation module that accommodates a data translation facility and a command generator.
The data translation facility supplies translation between a standard intermediate data storage format (ISF) and all application-specific formats; the command generator expands macro functions of a common command language to the specific command or key-stroke sequences for the application. These facilities were all added without altering the applications.

A central server interfaces with the application-encapsulation module for the appropriate data translators and command generators and activates the applications with the correct parameters. The user-interface behaves as a client, sending high-level instructions to the server. Data are transferred through the network by a standard file transfer protocol (Berkeley's remote copy: "rcp") and remote applications are initiated with the remote shell ("remsh") command. All applications employ the X11-server network communication protocol to output to a window on the workstation.

Both batch-oriented and interactive applications were encapsulated by the workstation. Interactive applications accept key-strokes from the command generator and display their response within a window on the workstation screen. The data-translation facility outputs results from batch-oriented applications in a standard file format; a graphical display facility grabs these results and shows them graphically.

Besides testing encapsulation techniques, a user-evaluation was started to analyze the response of clinicians to this type of graphical task support. In addition, other aspects, such as the reliability and validity of data from clinical information systems have been tested during the prototyping phase. Insight was gained into the set of tools required for data validation, inspection and analysis in the medical field.

3 Current related research

Research into methods for integration was predominantly focussed on integration of data in a network [4,5]. This research leads to environments that uniformly access data in different database management systems. Almost all research in this field focuses on schema integration: an integrated DBMS with a central overall schema that accesses the underlying databases. Dependent on whether that access goes via autonomous sub-DBMSs, the integrated DBMS is called a federated DBMS or a distributed DBMS. Heterogeneity is a special research issue that aims at accessing distinct database management systems from several vendors. To provide network-wide access, most DBMSs have a server that accepts (SQL-) queries from the network and returns the data. However, in general these servers are only accessible from database applications and their interfaces are not intended for foreign applications.

A second principal research topic is the separation of data access and management from the application. This approach seeks for generally applicable strategies to offer a loose coupling between data and application, allowing different applications to have a distinctive view on the data. The mediator concept of Barsalou and Wiederhold [6] is an attempt to separate the application from the database. An object-oriented layer hides the data access details from an expert system [7]. The objects once defined, exactly match the view of the expert system. This study showed that one can share the data among various applications, each application having its own mediator layer. The pursued approach is dedicated to the integration of one expert system and one database system on a single computer, and has no pretention to offer a general architecture for software integration.

In some ways, a generalization of this concept comes from the Helios consortium [8]. The basic architecture is built around a client-server concept, each client sending messages via a software bus to the server. Starting with a software bus, each application can put messages on the network with requests for data. Similar to a hardware bus, peripherals/applications can be plugged on the bus and receive instructions. Applications that contain the desired data, return it by sending the data object as a message back to the caller. The data message can contain any complex data object, ranging from textual data to images. New applications can be added to this bus by specifying its message interface in the Helios environment. The Helios architecture requires new applications to adhere to the Helios message structure. The technical layers show a sound concept, but their application is still restricted to individual tools rather than to an environment that provides task support for clinicians.

Hewlett Packard defines a Physician's Workstation as a graphical user-interface on top of an object-oriented DBMS [9]. This DBMS accesses data in various databases and offers a uniform access to e.g. a MUMPS database [10]. Alerts and tests are activated by the object-oriented DBMS by activating a knowledge base. This knowledge base contains a physiological model that tests for drug-drug interactions and drug-dosage. The architecture complies with open system technology: network messages are exchanged between the components and a broadcast message server routes the messages from caller to addressee. The focus of the system is mainly on data integration and the concept does not define possibilities for applications to address their functionality.

The National Library of Medicine (NLM) has initiated a research program designated as: Integrated Academic Information Management Systems (IAIMS) [11]. This program defines several different integration phases. Its initial phase covers institution-wide installment of a network. The main research objective is to extend the existing information systems with new functionality. Current research focuses on a central data entity dictionary [12] and the role of relational database technology in this integrated environment [13]. The Unified Medical Language System (UMLS) of the National Library of Medicine fits in this concept as the semantic link between the incorporated systems [14].

Greenes proposed [2] a new paradigm that is based on a heterogeneity of resources. He introduces "daxels", data access elements, that should hide the data access details from the applications. The Explorer-II system [15] addresses various access modes for knowledge resources in a coherent knowledge browse environment and exemplifies how heterogeneous knowledge resources can be interfaced in a user-friendly system. The approach followed is to adapt access of the resources to the Explorer-II specifications. The daxel approach is a logical continuation of Greenes' research to provide a multi-resource knowledge browser.

These examples demonstrate that the use of integration technology can have a large impact on the medical professional. To summarize, the following features can be distinguished in these systems: abstraction or a shift from procedural ("how") to declarative ("what") interfaces, combination of multiple data resources in one environment [5], addition of semantics to the database systems, and object-oriented techniques to obtain abstraction from access details.

4 Principal integration issues

Our concept of integration can be defined according to a five-layered model. Similar to the ISO seven-layered network-model reference, each layer is dependent on the lower layers and provides support for the upper layers. Our five-layered integration model fits into the OSI/ISO application network layer and defines at what levels applications can exchange information.

4.1 Defining OSI's 7th Communication Layer

The lowest layer consists of hardware integration or physical integration. At this level, various computers are connected in a network and behave as virtually one machine. The components may run different operating systems with different file systems, but to the application the environment acts as virtually one system; remote files are addressed similarly to local files and remotely executing applications are handled as if they run locally.

The second level of integration treats the different data sources uniformly: although scattered over various database management systems or file systems, the data can be addressed as having one format and being stored in one file. Differences in data organization and representation are transparent to the upper layers.

The third integration layer deals with software or functions. All functions available in a network environment compose one large virtual application. Workstation applications and users can address the various functions to data without having to know the typical commands to call that specific function.

The fourth layer of integration links the different user-interface styles of the integrated applications and presents the user with a uniform interface. All user-interfaces are available from one screen at the same time.

The top layer shields the semantic heterogeneity of the integrated components. A semantic model unifies the different meanings and nomenclatures in one universal model.

Ultimately, this is the model for total integration of existing resources. This multi-layer integration model is designed around a number of elementary features as discussed in the following sections. The model is shown in Figure 2.

4.2 Connectivity
Data, knowledge, and functions are regarded as the resources for a medical workstation. The medical workstation defines a uniform view on these resources, which are inter-connected. All integration layers support this connectivity issue. At the lowest level, a network connects the resources when residing on different hardware. The network connection consists of a medium and a protocol. The network protocol is required to provide high-level support for resource connection and be operating system independent.

For connecting data and function resources, the Internet protocol can be used; for connecting the user-interfaces to one display the X11 client-server network provides a sound multi-vendor standard protocol. For those components missing software that supports these protocols, a protocol-convertor has to be defined. For example, an Ethernet-RS232 protocol-converter acts as a connection with more traditional RS232 network environments, and an X11-converter to translate interface requests to equivalents for an X11-server.

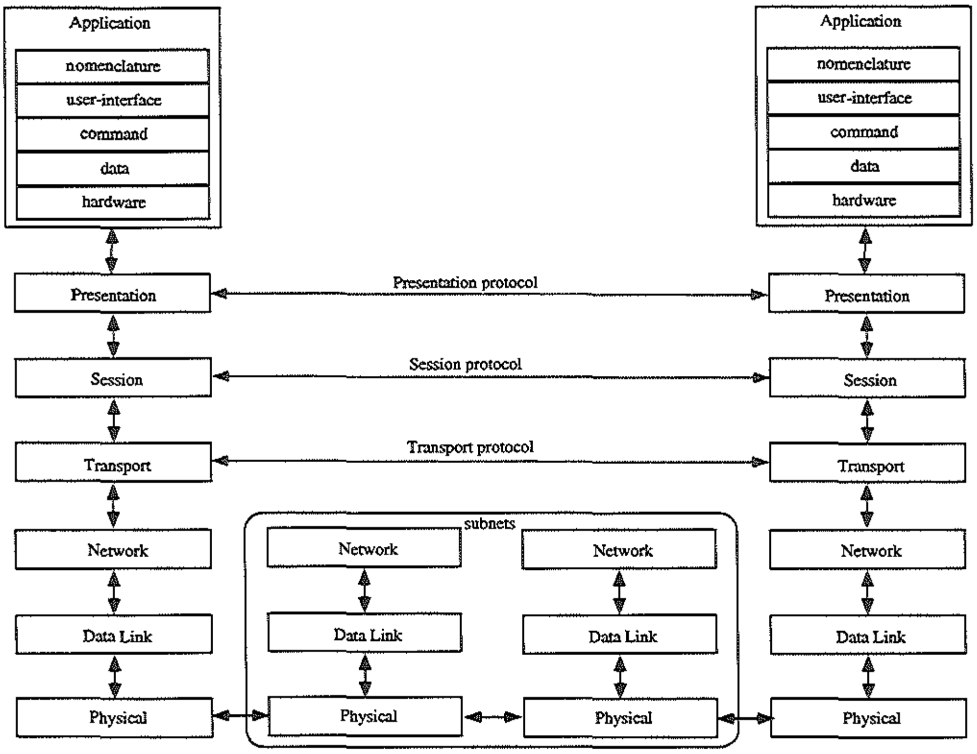


Figure 2. The location of the Integration Model in the OSI Communication Reference Model.

To master the complexity of dissimilarities between the interfaces of the existing resources, a common interchange language has to be defined. This language contains syntactical constructs for addressing data, functions (including data presentation functions), and knowledge. Although some standards evolve for data communication among different resources [16], they principally lack capabilities for activation of functions and triggering of knowledge. The interchange language has primarily to be founded on standards, such as SQL for accessing data, and HL7, Medix or Edifact for transferring data. In addition, access to new resources has to be easily assimilated as part of the language. Each resource is equipped with an interpreter that accepts instructions of this general interchange language and translates them to the resource-specific instruction(s). Results are translated from the resource format to the general interchange format.

The connection of data from different databases requires additional research; when linking records from several databases in order to perform a multi-database join, one should be able to specify a medically-meaningful time-orientation for the join; one is not interested in performing an ordinary Cartesian join based on a key (linking each record of resource A with all corresponding records of resource B), but only in those joins where a record of resource B is in a certain time interval of a record of resource A [17]. The interchange language should contain commands for specifying this medically-oriented record linkage. Similarly, the join should normalize to a particular nomenclature (fifth integration layer): complementing the missing data fields, committing normalization functions, and checking for validity and/or plausibility of the values are implicitly associated with the join operation.

4.3 Modularization

Modularization has been a software development issue for many years. When developing large systems, one divides the system into independent modules and for each module an interface is defined hiding its implementation. These modules can be developed and tested separately from the others and can be linked together in the final system. The interfaces are in general composed by a set of functions that can be executed. The various modules are available as libraries and are linked at compilation time into the final system. Each modification in a module requires the system to be linked again. This modularization focuses on the development aspect and is not valid during run-time. To extend this modularization notion, dynamic linking libraries (DLL) have been invented: at application load time, the associated libraries are identified and also loaded. At run-time, changes to the library have no immediate effect.

For the medical workstation, the level of modularization should go beyond what is offered now, by dynamic linking libraries while many of the existing resources are not available as libraries but as autonomous entities. In addition, many applications can use the same resource at the same time, and consequently changing a resource requires all related applications to restart. The distribution of libraries over different processors remains a topic for research and has not been solved satisfactorily [18].

A commonly used approach to exchange information between autonomous resources that complies with the modularization issue, is the message mechanism. Each resource accepts a specific set of messages and returns results as messages to the caller. The messages are expressed in a common interchange language. Changes to resources can be effected directly in this environment without modifying the other resources as long as the resource obeys the same set of messages.

4.4 Shareability

In the previous sections, it was tacitly assumed that resources are dispersed in a network. The notion behind this assumption is to leave the central system concept where all resources are available on one machine and to focus on a computing-network that appears to be virtually one personal machine. From this network approach it is evident that the resources should be shared among the various users in the network. Each user should - if permission is granted - have complete access at all times to all resources supported in the network.

This network approach supports the integration of resources that demand a variety of specific hardware, operating systems, or file systems. Through the network, these resources are accessible from any personal workstation. The advantage of this approach is that one can distribute the different resources around various processors, allowing to take full advantage of both the specificities of the hardware and the available processing power. Each resource can be assigned to a particular personal workstation. Addressing a resource activates a remote processor and in this way distributed processing can be obtained.

Shareability does not necessarily imply the absence of private resources. Although these private resources should comply with the notion of sharing, they might require a password to accompany a message. Only clinicians who have a password may have access to the specific resource.

4.5 Extensibility

In order to take full advantage of the latest developments in software development, the workstation should be capable to be dynamically extended with new or changed resources. As described in the section on modularization, unless the message definition changes, the modularization constraint abstracts from any change to the intrinsics of the resources. Extensibility covers integration of new resources not previously contained and modifications of the message definition.

A well-known strategy to cope with a dynamically changing environment is the indirect referencing mechanism: applications do not directly forward messages to the destination resource, but send the message to a system-wide message router [19]. The router consults a database and fills out the name of the resource and the defaults if necessary. The clinician can specify his own private defaults to overrule the system defaults. This indirect referencing allows the router to use new resources without changing the application. For instance, in addition to a t-Test message of the statistical package BMDP, a t-Test message of SAS can be defined. The router decides which resource (SAS or BMDP) is going to resolve the question. This decision is based on the user's preferences, the load of each resource and the other features. Each new resource can be inserted in the router's database with a special resource editor.

4.6 Encapsulation

Having outlined a number of features essential to the resources, the question arises how existing resources can conform to these specifications without adapting the code. The remedy is called encapsulation. With encapsulation, a front-end is attached to existing resources that decodes the standard interchange messages into resource-specific instructions or data. This front-end layer incorporates in general two modules: a data translation part and a command generation part. The data translator accepts messages containing data and rewrites the data in the format acceptable by the resource. Output from the resource is intercepted by the data translator, composed into a valid reply message and returned to the caller. The command generator accepts messages that specify a function and expands them into a series of commands or key-strokes for the resource.

The workstation environment should provide tools for building these data translators and command generators. For data translators, one could think of a tool generating a translation prescription for the UNIX "awk" data processing editor. Two approaches for command generation or macro expansion exist. Firstly, one can use a macro recorder to record keystrokes. To each registration, a high-level name can be assigned. Specification of this high-level command in a message causes the macro-recorder to play the associated keystrokes. Each macro script may contain variables that are bound at run-time. These variables have to be specified as part of the message and are filled out by the macro expander.

The second approach is more lexically oriented, and consequently more suitable for extensive command languages. The command structure is defined as a grammar, which describes how high-level functions should be expanded into elementary instructions for the application.

4.7 User-friendliness

The advent of graphical workstations enables to offer clinicians a friendly user-interface. Two aspects of friendliness are described here: (1) the control and organization of the user-interface and (2) the medical model behind the workstation.

The graphical environment facilitates display of signals, images, and text on one screen. A windows system better suits the concept of browsing through a network of data resources. Initially, the most relevant data for a specific clinical task should be displayed. Successively, the clinician can decide to acquire more detailed information from a certain resource. A windows environment provides new pop-up windows showing these additional data and refrain the initial window from updates. With a mouse pointing device, one can zoom in on data and move the various windows. All functions can be designated with the mouse, with a minimal use of the keyboard.

The MW2000 workstation project has chosen the X11 OSF/Motif user-interface as its typical window environment. All hardware should administer an X11 server for presentation of the results on the screen. For MS-DOS PCs, X11 servers are available under Windows 3.0. In this way, the X11 user-interface protocol is the most common approach that is currently available on an extensive range of hardware.

Initially, the user-interface uses a medical model for its decision on what to display and how to expand data. This model can be customized for each user and contains a comprehensive data model for all available data. All resources have to inquire this medical model about data-specific information. For each data this model contains its location, pretty name, type, code-list, plausible and possible ranges, synonyms, relations with other data, and so on. The model can graphically be edited by a clinician and is applied for modelling terminology and data in the area of congenital heart diseases and andrology [20,21].

Although more aspects of friendliness can be defined, it is thought that a system fulfilling the requirements outlined in this paper could serve as a basis for further research in this field. Current experience with task-oriented user interfaces for medical professionals is limited by the fact that no system is yet able to support a complete task. Our prototype user-evaluation has shown that covering these two basic aspects of user-friendliness is already a giant leap forwards.

5 Potential pitfalls and problems

In the previous sections, the key issues for a new integration architecture have been outlined, together with some of the most important related requirements. In this section, the technical problems involved with this approach will be discussed, together with their potential solution.

5.1 Object-oriented management of complexity

Typically, a workstation is attached to a realm of resources. One of the first problems is to create structures for managing complexity. The workstation should offer both an easy-to-use environment for the clinicians and an advanced platform for new software developments. The object-orientation paradigm is applied to simplify access and management of the resources [22].

Each resource, i.e., application, database, or knowledge base is represented as an object. This object hides the access to private data and private functions from the outside world. Objects that shield the same type of resource are aggregrated in a class object. Besides a private part, it also contains a public part that contains functions and data. The public functions and data can be accessed by sending messages to the object. These messages specify what public function or data should be accessed. All objects have the same message structure. Existing applications have an object associated that specifies which functions and data are accessible for the application. These functions are high-level macro functions that are expanded by a command generator to the actual application instructions. When accessing the public data, a data translator will take care of translating the data from the application-specific format to the workstation's internal storage format of the data.

Class objects also accept messages but route the message to one of its children. The decision to what child is based on information from the medical data model. The class object inquires the medical data model for specific information and uses this to route the message. For instance, all objects that shield data belong to a particular database object. All database objects are children of a multi-database object. Messages requiring data can be sent to this multi-database server. This server will then query the medical data model for the right database server. The message will then be routed to the appropriate object that shields data for that particular database management system.

The operator-overloading feature also minimizes the complexity for the workstation applications. Different resources or objects can respond to identical messages. For instance, the objects associated with the statistical software packages BMDP and SAS both accept a message called "t-Test". These messages are not application-specific, but are specified as a general message. Consequently, the message scheme allows a more declarative ("what") attitude and leaves the transformation to a procedural approach to the object associated with the application.

5.2 Idiosyncratic messages with different meaning

The feature of operator overloading introduces the risk of ambiguity: idiosyncratic messages that have a different meaning. The router uses the message name to look up the resources that accept that message. Consequently, the meaning of the message should be unambiguous. This ambiguity can be avoided by having a tool for registration of the messages. This tool checks for consistency and ambiguity of the messages.

Ambiguity in the message fields can be handled easily. The interpretation of the message fields is performed by the object (server); this provides an interpretative context for the meaning of the field.

Not all message fields have to be specified. Three types of fields are discriminated: (1) essential fields, i.e., fields that have to be specified by the caller, (2) necessary fields that have to be filled up either by the caller or by the router, and (3) additional fields that can be specified but are not necessary.
The router checks messages for completeness, i.e., essential fields, against information stored in the database and additionally, it adds all necessary fields not specified with the associated default value. For appending the necessary fields, the router inspects the user's default settings and preferably uses this default. Otherwise, it uses the default values specified in the system's database.

5.3 Reliability and performance

While the architecture is based on a network-wide message-passing scheme, it is essential that the network protocol used provides a reliable point-to-point communication and that failures and errors are detected and reported. Evidently, in a medical environment, it is essential to discriminate between data not being registered and those temporarily not available. The Internet protocol provides a reliable point-to-point connection with error-reporting facilities. In addition, the protocol also provides synchronous (i.e., messages arrive in order) and asynchronous connections at the same time.

The performance of the workstation is dependent on its reliability: a high retransmission rate reduces the performance [23]. Worse even, if a particular task depends on a number of resources, a high connection refusion will prohibit a successful termination of the task. Experience with the prototype has shown that our network topology is reliable enough and provides an acceptable performance.
In order to improve the availability of resources, it can be decided to incorporate redundant resources in the network. The message-routing algorithm could detect connection refusions and assign an alternative resource automatically. Consequently, attention should be paid to synchronization of the resources. For function resources, the possibility of introducing redundancy is easier to control than for patient-care databases.

In addition to the supervised introduction of redundancy, some resources might have a coincidental overlap. Class objects might use this feature to gain a better performance. Firstly, the response time varies among the resources. The class manager could benefit from this when alternative resources are available, by addressing the resource with the shortest response time. Secondly, series of requests can be combined to a single request when all individual requests are resolvable by one resource. Again, this might imply an abduction from the resources that are used by default, and the user is requested to specify his preferences related to the procedure. Attention should be paid to the fact that routing decisions directly influence reaction times of the resources. The class object routing algorithm should dynamically consider the alternatives and continuously measure the response times.

Although one can take advantage of this coincidental overlap between resources, caution should be paid to the following caveats:
1. When data resources have an overlap in the variables stored, one needs to be sure that the populations covered in the data resources are identical and that the registered variables contain the same information. Stated differently, results should be repetitively obtained whatever the consulted resources are. Concerns about quality of data might have an influence on the decisions what to treat as overlapping resources.

2. Dead-locks might occur in the access of resources. This is only the case when resources can be locked for access. Dead-locks can be solved by providing each resource with an automatic release feature: after a certain lock time, the application that seized the resource is requested to unlock the resource. In general, resources are only locked during the execution of a message. As soon as this processing terminates, the resource is released and other messages can continue.

3. Attention should be paid to the overhead caused by the message mechanism and the message router. An alternative approach for the message router would be to return the name of the resource capable of solving the request to the sender of the message. This sender could then start directly communicating with the resource.

5.4 Security and protection

As already discussed in the section on shareability, in principle all resources can be addressed by all users. Access, however, can be restricted to private resources by requiring a password. Each private resource should have its own password-checking procedure. There are two mechanisms for obtaining a password: (1) by requiring a password field in the message, and (2) by interactively asking the user for his password. The latter mechanism can be extended to ask a login name as well.

When sending messages containing private data, an encryption procedure can be applied to the data contents. Both the caller and the target resource should agree upon the encryption key used.

5.5 Synchronization

Each application sending a request for data, gets a local copy of the data. For some of these applications, it is essential that they keep track of the updates of the original data. This feature, i.e., synchronization of data, is also referred to as hot-linking. Each application can enable hot-linking by processing update messages for those particular data. Each data resource sends updates of the data to the router when modifications are applied. The router then broadcasts the updates to all open (i.e., connected) applications and resources. Consecutively, each application and resource decides to accept or discard the update message.

In general, the router will start an application when an initial message arrives for the application. Consequently, a user might start an application several times, resulting in a series of windows on the screen. However, each user might specify per application how many open application windows are allowed on his desk-top; a decision to restrict the number of identical open applications to one user forces the workstation to use existing open application windows, a zero value disables the application to appear on the user's desk-top. For example, this feature can be used for an application that displays a histogram, and forces update requests to be directed to the same histogram application, resulting in an update of the open histogram window.

The workstation also contains possibilities for editing data. To avoid changes in the local copies of the data, most of these editors compose an update message for the original data resource. This message is then sent, the data resource updates the data, and the data resource sends an update message to all open applications, including the editor. The editor facilitates processing of this update and displays the changed data. This scenario shows that the features discussed in this paper, provide a clear approach of editing local copies of data.

All data resources return data upon request in a message format. However, the return message can be read by the application in a stream mode. This means that not necessarily the complete message has to be read before starting processing. The stream mode delivery of the data prevents storage of datasets that may be too large for the workstation disk. This stream mode processing features parallel processing, i.e., one process collects the data and puts it on the network while simultaneously another process for retrieves the data from the network and processes it.

6 Putting the modules together: a hypothetical example

In the previous sections, the basic features of the workstation have been presented together with solutions for some of the problems. In this section, the object-oriented approach will be outlined in the context of an example. From the prototype for clinical research, the following example can be presented. A clinician wants to have a clinical overview of all patients that had a sudden death. The new architecture should then provide him the following support.

1. The user has to log in the system. A graphical desk-top is initiated.
2. He then selects the task he is interested in, in this case clinical research.
3. Accordingly, the user-interface renders him with all the functions able to perform, as well as a collection of queries designed in previous sessions.
4. The clinician decides to create a new query. The user-interface generates a message "select data" and sends it to the router.
5. The router searches its database to locate a service that accepts this message. It decides to forward the message to the data selection service.
6. Before actually forwarding it, the router activates the data selection service if it was not yet accessible.
7. The data selection service accepts the message and shows an output window on the display of the user.
8. At the same time, the data selection service sends a message to the medical data model service (again by way of the router) to obtain the clinician's data model.

The model contains a hierarchy of concepts like clinical overview, risk factors, laboratory data etc. The terminal tree nodes are the actual data stored in the various databases (Figure 3).

The clinician selects the node 'clinical overview' of the data-model tree in the data-selection service. This node is the top node of a subtree, containing all the data that correspond with a clinical overview: risk factors, laboratory data, other patient information such as history data, etc. The clinician specifies a search criterium and affirms the selection. Succeedingly, the data-selection service composes a "select" message for the multi-database server (via the router) with an SQL-query as its argument, containing the variable names, their addresses (relations and databases) and the search clause. The multi-database server will subsequently disseminate the query into SQL subquery messages for all database management systems involved. Responses from these subqueries are joined by the multi-database server and returned to the data selection server. The data can be saved by sending it as a message to the data manager (this service might actually reside on a different host). The clinician might start the graphical presentation service, select the named data set and interact with a histogram settings window, select the variables and specify groups for the histogram, pie chart or other graphical presentation. To obtain a print of this graph, the graphical presentation service forwards a message for the print service containing the necessary data. Finally, the clinician can save the system's state (for next sessions) and log off.

7 Conclusions

In this paper, the key issues for an integrated medical workstation have been described. Current experience with a prototype medical workstation taught that the technical provisions for this architecture are available; both network technology and graphical display systems have evolved to reliable components for a medical workstation. The object-oriented approach, realized in a client-server scheme, enhances new systems with a modular, shareable and extensible platform. Moreover, the declarative style of questioning resources shields both clinicians and applications from access details and specificities. This declarative approach reduces the complexity and defers the actual work to the resource service itself. Supplementary, existing applications can be encapsulated in order to obey the standard interchange language defined for the workstation. A central message router uses a database to assign resources to a particular request. New applications/resources can be dynamically added in the router's database and other resources do not require updates before being able to take advantage of this new resource.

Although the proposed architecture is more related to the technical layers of integration and less explicit about semantic integration, experience with the prototype learned that this approach is already valuable for novice users. Furthermore, this architecture can be viewed as a powerful operating system for integration. Although the basic elements are covered for providing a truly task-oriented support for clinicians, it is recognized that new research projects should be started to see how clinical contexts can best be defined and to investigate what parts and how the medical data model can be edited by end-users. However, before being able to primarily focus research on task support, it was felt to be essential to offer the tools for integration and acquire a new vision on software components.

Part of this new architecture has already been realized in a next generation workstation. Both the message router, the medical data model, and various database services have been established. Current developments focus on the incorporation of data analysis services.

Acknowledgement

Part of this work has benefitted from a grant of Hewlett-Packard Medical Products Group Europe.

References

[1] GREENES, R.A. Promoting productivity by propagating the practice of "plug-compatible" programming. *In* "Proceedings of the 14th Symposium on Computer Applications in Medical Care" (R. Miller, Ed.). Vol. 14, pp. 22-26. IEEE Computer Society Press, Washington, DC, 1990.

[2] POWER, L.R. Post-facto integration technology: new discipline for an old practice. *In* "Proceedings of the First International Conference on Systems Integration". Vol. 1, pp 4-13. IEEE Computer Society Press, Morristown, 1990. p4-13

[3] VAN MULLIGEN, E.M, LANGHOUT, A., TIMMERS, T., LEAO, B DE FARIA, AND VAN BEMMEL, J.H. Software integration on a medical workstation. *In* "Proceedings of the IMIA Working Conference on Software Engineering in Medical Informatics". pp167-179. North-Holland Publishers, Amsterdam, 1990.

[4] LANDERS, T.A., ROSENBERG, R.L. An overview of multibase. *In* "Distributed Data Bases" (H.J. Schneider, Ed.). North-Holland Publishers, Amsterdam 1982.

[5] BERTINO, E. Integration of heterogeneous database application through an object-oriented interface. Inf. Sys. 14, 5 (1989).

[6] BARSALOU, T., AND WIEDERHOLD, G. Knowledge-directed mediation between application objects and base data. *In* "Proceedings of the Working Conference on Data and Knowledge Base Integration" October, 1989.

[7] WIEDERHOLD, G., BARSALOU, T., SURJANSKY, W., AND ZIUGMOND, D. Sharing information among biomedical applications. *In* "Proceedings of the IMIA Working Conference on Software Engineering in Medical Informatics". pp. 49-64. North-Holland Publishers, Amsterdam, 1990.

[8] DEGOULET, P., COIGNARD, F.C.J., LAURENT, M.C., et al. The HELIOS european project on software engineering. *In* "Proceedings of the IMIA Working Conference on Software Engineering in Medical Informatics". pp. 125-137. North-Holland Publishers, Amsterdam, 1991.

[9] TANG, P.C., ANNEVELINK, J., FAFCHAMPS, D., STANTON, W.M., AND YOUNG, C.Y. Physicians' workstations: integrated management for clinicians. *In* "Proceedings of the Fifteenth Annual Symposium on Computer Applications in Medical Care", pp. 569-573. McGraw-Hill, Washington, DC, 1991.

[10] ANNEVELINK, J., YOUNG, C.Y., AND TANG, P.C. Heterogeneous database integration in a physician workstation. *In* "Proceedings of the Fifteenth Symposium on Computer Applications in Medical Care", pp. 368-372. McGraw-Hill, Washington, DC, 1991.

[11] LUNIN, L.F., AND BALL, M.J. Perspectives on Integrated Academic Information Systems (IAIMS). J Amer Soc Inf Science. 3, 39 (1988).

[12] CIMINO, J.J., HRIPSACK, G., JOHNSON, S.B., FRIEDMAN, C., FINK, D.J., AND CLAYTON, P.D. UMLS as Knowledge Base - A Rule-Based Expert System Approach to Controlled Medical Vocabulary Management. *In* "Proceedings of the Fourteenth Symposium on Computer Applications in Medical Care", pp. 175-179. IEEE Computer Society Press, Washington, DC, 1990.

[13] JOHNSON, S.B., FRIEDMAN, C., CIMINO, J.J., CLARK, T., HRIPSACK, G., AND CLAYTON, P.D. Conceptual Data Model for a Central Patient Database. *In* "Proceedings of the Fifteenth Annual Symposium on Computer Applications in Medical Care", pp. 381-385. McGraw-Hill, Washington, DC, 1991.

[14] MCCRAY, A.T. The UMLS Semantic Network. *In* "Proceedings of the 13th Symposium on Computer Applications in Medical Care", pp. 503-507. IEEE Computer Society Press. Washington, DC, 1989.

[15] GREENES, R.A. "Desktop knowledge". A new focus for medical education and decision support. Methods Inf. Med. **28**, 332 (1989).

[16] OSTLER, D.V., HARRINGTON, J.J., HANNEMYR, G. A common reference model for healthcare data exchange - p1157 medix system architecture. *In* "Proceedings of the Fourteenth Annual Symposium on Computer Applications in Medical Care", pp. 235-238. IEEE Computer Society Press, Washington, DC, 1990.

[17] ROOS, L.L., AND WAJDA, A. Record linkage strategies. Methods Inf. Med. **30**, 117 (1991).

[18] CHIN, R., AND CHANSON, S.T. Distributed object-based programming systems. ACM Comp. Surveys. **23**, 91 (1991).

[19] GAGAN, M.R. The HP softbench environment: an architecture for a new generation of tools. Hewlett Packard Journal. **41**, 36 (1990).

[20] VAN DEN HEUVEL, F., TIMMERS, T., VAN MULLIGEN, E.M., AND HESS, J. Knowledge-based modeling for the classification and follow-up of patients with congenital heart disease. *In* "Proceedings of Computers in Cardiology Conference", pp. 737-740. IEEE Computer Society Press, Venice, 1991.

[21] TIMMERS, T., VAN MULLIGEN, E.M., AND VAN DEN HEUVEL, F. Integrating clinical databases in a medical workstation using knowledge-based modelling. *In* "Proceedings of the Seventh World Congress on Medical Informatics", pp. 478-482. Elsevier Science Publishers, Geneva, 1992.

[22] STEFIK, M., AND BOBROW, D.G. Object-oriented programming: themes and variations. The AI magazine. pp. 40-65, 1985.

[23] TANENBAUM, A. Computer Networks. Prentice-Hall Inc., EngleWood Cliffs, NJ, 1981.

CHAPTER 7

A New Architecture For Integration Of Heterogeneous Software Components

E.M. van Mulligen[1,2], T. Timmers[1], J.H. van Bemmel[1]

[1]Dept of Medical Informatics, Erasmus University, Rotterdam, The Netherlands
[2]University Hospital Dijkzigt, Rotterdam, The Netherlands

(Methods of Information in Medicine 1993;32:292-301)

Abstract

An architecture is described that integrates existing applications in a network-wide system. The architecture follows the new open software paradigm, and defines kernel and application services that collaboratively solve the tasks of end-users and provide them with an intuitive user-interface. This paper describes the message language and the kernel mechanism for addressing application services. The architecture has been developed as much as possible to conform with current standards.

keywords: integration, client-server, connectivity, workstation

1 Introduction

The appearance of mature network technology and standardized graphical user-interface systems has not offered much benefit to the professional end-user. Notwithstanding the introduction of networks and powerful workstations, the benefits remain limited: a tradition of isolated, stand-alone applications that ignores the network and bypasses any present graphical user-interface system limits this technology to the operating system level only: file transfer, terminal sessions, sometimes remote execution, and a graphical user-interface that is used for multiple terminal windows on one display only. In addition, many current software applications do not provide standardized data import and export facilities and offer only the software elements necessary to fulfil a single task. Consequently, this demands that the domain expert be a computer expert as well, and to master all applications required for solving his questions.

1.1 Prototyping an integrated medical workstation

Starting with the development of the prototype of an integrated medical workstation for the support of clinical research, the project has been directed towards how network technology and graphical user-interfaces can be applied to existing applications, to obtain an integrated environment that maximizes the support for the clinician [1]. User validation was carried out to test the benefits for the clinician in a typical and wide application spectrum such as clinical research, i.e., a variety of different database management systems, statistical applications, presentation software, and word-processors.

1.2 Integration reference model

The integrated medical workstation defines an integration reference model with five integration or abstraction categories:

(1) *Hardware category.* A hardware and operating system integration category hides the differences between addressing local or remote files, and between local or remote execution.

(2) *Data category.* The differences in data storage format are abstracted on this level. The user can access all data from within each application. The data are automatically translated to the application-specific format.

(3) *Command category.* The functions of all applications can be accessed through a macro command language. This language is uniform for all applications and frees the user from knowing the application-specific instructions. This macro language

is translated by an expander to the appropriate application instructions, either keystrokes or batch commands.

(4) *User-interface category.* This category consists of two aspects. Firstly, each application is presented in a separate window on the display. Secondly, each application uses the properties of the user-interface system to interact with the user.

(5) *Nomenclature category.* The nomenclature category abstracts from the application-specific vocabularies.

1.3 Integration issues

Experience with a prototype medical workstation has been used as input for the definition of a new, more general, workstation architecture for an integrated workstation as follow-up of the prototype that supports the integration categories as defined above. This new medical workstation architecture has been defined with a series of issues in mind: *connectivity, shareability, modularization, user-friendliness, extensibility, and encapsulation.* These aspects are briefly outlined below, followed by the general outline of the integrated workstation architecture.

1.3.1 Application integration paradigm

The new vision on software no longer considers applications as a complete, closed set of functions that run in isolation from other applications, but rather as a software component that contains some new functions and refers to functions of other components. Typical functions such as file access, printing, help, etc. are not contained as a function in each separate software component, but are accessible in one component that is shared by many other components (*shareability*). This approach is often referred to as open distributed processing, or as group-ware [2,3]. Referencing functions of other components demands a standardized mechanism to deliver parameters and data. These software components comply with *open software technology*, and each application can hook into these components to have a particular function performed.

Typically, these software components can be dispersed through a network, offering the possibility to assign software components to computers that are best suited for that task: computers with a huge mass storage contain data-intensive software components, whereas statistical and computational components are available on number crunchers. User-interface modules typically run on workstations or PCs. This network-wide distribution of components requires each software component to be able to connect to other software components via the network (*connectivity*).

One of the research aims of the medical workstation project has been the combination of this new vision on software with existing software. Typically, the medical application domain contains a range of applications (hospital and departmental information systems; laboratory, radiology and pharmacy systems; systems for analysis of ECG, EEG, etc.; systems for patient monitoring; image processing systems; statistical packages; decision-support systems; etc.) that cannot easily be replaced by software components that obey the new open software paradigm. Moreover, to make this integration architecture successful, a critical mass of applications has to be available as software components within the network. Consequently, the maintenance and support of such an environment would be immense and clearly diminish the advantages of this integration architecture.

Software components should be inter-connected according to the client-server model. The client software component composes a request and delivers this request to the server. The server will respond to this request and return a message containing the results for the client.

To re-use existing applications and incorporate them as open software components in the workstation, an *encapsulation* layer has been introduced. Each software component has its own layer that converts the request to an application-specific data format. For instance, references to functions are expanded to a series of keystrokes or batch commands that are specific for the application. Output of the application can be shown interactively or written to a file. The Yanus project at our department [4] aims at developing a methodology for analyzing both graphical output and output files, and translating them to a standard output.

Management of complexity

The inter-module dependencies increase the complexity of management. To minimize this management, two mechanisms are incorporated. The first mechanism is that the communication between components has been restricted to contain only declarative information (specifying "what") and requires the accepting software component to translate the declaration into a procedure ("how"). Secondly, no components are required to specify the target software component when addressing foreign functions. The system can consult an accessors database [5] to search for a software component that can solve the request. New components can be added by editing the database.

This database accommodates for each user group, i.e., manager, clinician, statistician, developer, and demonstrator, and for each individual user separately the components that can be accessed. Through this database, different views can be defined for the various users. The user-interface is provided as a separate component in the architecture. For each user group, a user-interface component can be created and tailored to the specific tasks of that group.

Issuing commands

The integrated workstation offers the users an environment in which they can request any function to be applied to data anywhere in the network. These requests can be sequentially stored as a program that can be invoked via the user-interface. For example, a clinician might define a data collection, analysis and presentation program that allows him to quickly obtain a daily report about the status of his patients. These macros can be obtained with a macro recorder and stored as workstation programs.

Another type of support is the medical data model available for the clinician. In this model, all data available from the workstation have been described, together with their inter-relationships. This data model contains a hierarchy describing the data (Figure 1a,b). The data model contains the mapping from the data terminology provided to the user for the specific variable names of the database(s).

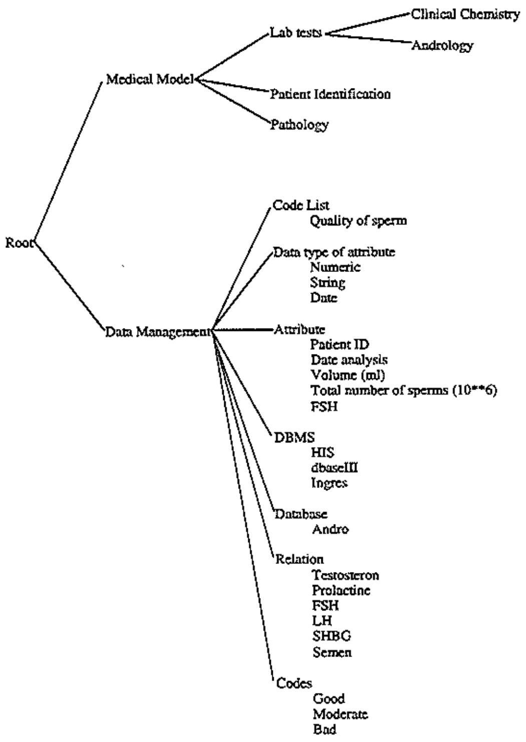


Figure 1a. Overview of an example data model for andrology. The concepts are organized into two main branches: one for the medical data model as presented to physicians, and one for the data management. For some concepts, instances are shown in a list below the concept name (e.g., concept Attribute with instances Patient ID, etc.).

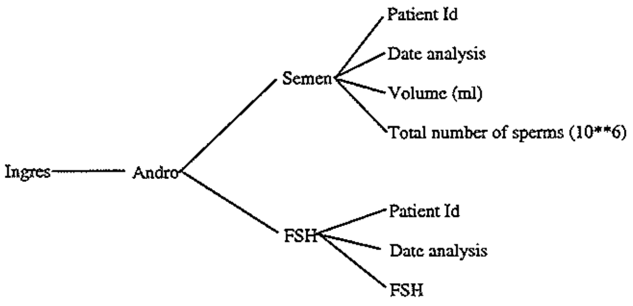


Figure 1b. Linking between the instances. Links is_a_database_of, is_a_relation_of and is_a_attribute_of are shown for the database management system Ingres. From within the medical data model, links are defined to the attributes that are related to the medical concepts.

Software components can have their own user-interface for interaction. All components use a similar user-interface management system with a typical style guide. Other components that do not have a graphical user-interface of their own, can address special components that provide user-interface functions.

1.4 Related research projects

Some of the issues outlined above have also been addressed by others. The Integrated Academic Information Management System (IAIMS) facilitates data access to all existing information systems from each workstation [6]. The primary focus is the development of an institution-wide network that inter-links all systems, and a methodology to access these information systems. Future developments will focus on the integration of data with applications.

Hewlett Packard's Physician's Workstation covers integration at all categories described above [7]. However, the main interest is the integration of databases; incorporation of computational software components has not yet been addressed. A Broadcast Message Server has been implemented to match messages with message patterns stored in the BMS database. Associated with each pattern is an application that can handle the message.

The Helios project utilizes a software bus to achieve integration [8]. A mechanism exchanges messages between the various components. However, these components are limited to those that are specially developed for interaction with the software bus.

Barsalou and Wiederhold [9] limit their integration to the data category. Their main interest is to investigate abstraction mechanisms for access to a database by so-called mediators or view objects.

The Explorer-II project of Greenes is an example of how different knowledge components can be combined into a consistent system [10]. This research focuses on the different knowledge access modes. Currently, one subject of research is on "daxels", data access elements, as an equivalent of the mediator concept [3].

The Field software development environment [11] connects software development tools. It supports the exchange of messages between the existing and new tools. The protocol used to forward messages from a client to a server process is selective broadcasting. All messages are forwarded to the message server, and this server will direct the message to all services that have a message template registered that matches with the message. Before one of the tools is started, it will notify the message server about the messages in which it is interested.

Other researchers have investigated the use of an open distributed programming environment. This environment can be used to distribute applications around various processors, and facilitates inter-connection among the application modules [12,13].

2 Methods

2.1 Prototyping

To test the integration concepts outlined above, a prototype was developed to support clinical research, directed at integration for all categories. The prototype consists of one central server that addresses all applications and data. This server accepts (macro) instructions from a user-interface and activates the corresponding scripts. Those scripts transfer data through the network and activate functions in components. The communication was performed on the operating system level with Arpa-Berkeley's file transfer protocol (ftp) and remote execution (rexec).

The primary aim of the prototype was to test whether the support for users was improved by an integrated environment. A user evaluation was designed addressing six clinical research problems, to be solved with the prototype. After a one-hour introductory tutorial, most clinicians were able to solve all problems correctly within two hours. Their response to the system was mainly positive and they thought the system to be useful.

Both the input from the user evaluation and the experience with the development of the prototype resulted in a refinement of the integration architecture. This architecture includes multiple services and communicates on the application level rather than on the operating system level. Furthermore, the user interface has been completely separated from the functionality of the workstation architecture, allowing each user group to have their own view on the functionality.

2.2 Concepts

The communication between software components can be seen as functioning on the seventh layer (application layer) of the ISO/OSI communication reference model [14]. Each component communicates with another component through the network. The communication between the components is expressed by messages. Each component behaves like a server: any message on the network that is addressed to the server is removed from the network, analyzed and executed by the server. Results are returned to the component that composed the message. In this way, the network software provides a synchronous, reliable point-to-point communication, and messages arrive correctly at the server. For the workstation, the Berkeley socket mechanism has been selected as being the most widely used Internet network protocol that runs under TCP/IP with Ethernet.

2.2.1 Server

Each server contains a wait loop for connection requests. Upon acceptance of a connection request, the server forks a new process that creates a new socket connection with the caller. Through this socket, messages can be exchanged. Messages can be obtained by reading from the socket, and messages can be returned by writing on the same socket. A server can, in turn, also act as a client and send requests to other components.

2.2.2 Client

When a client wants to address a function in a foreign component, it first sends a connection request. If the connection is confirmed, a socket descriptor is returned through which messages can be passed to the server. Subsequently, the client composes a message and writes it onto the socket. The connection mechanism is depicted in Figure 2.

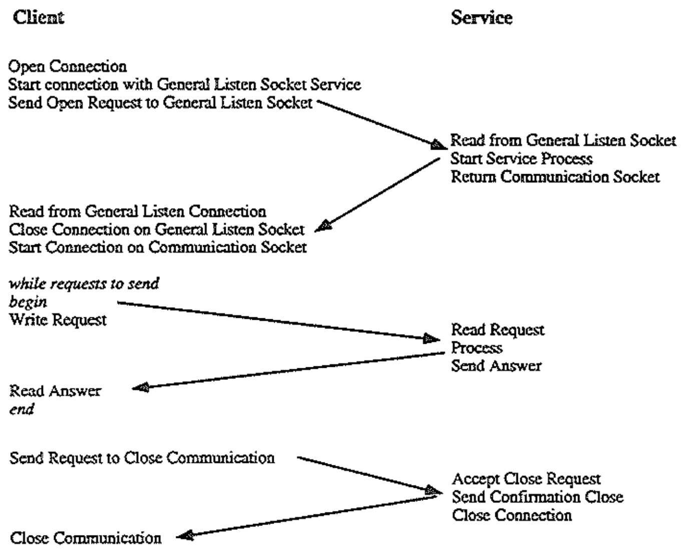

Client
Service
Open Connection
Start connection with General Listen Socket Service
Send Open Request to General Listen Socket
Read from General Listen Socket
Start Service Process
Return Communication Socket
Read from General Listen Connection
Close Connection on General Listen Socket
Start Connection on Communication Socket
while requests to send
begin
Write Request
Read Request
Process
Send Answer
Read Answer
end
Send Request to Close Communication
Accept Close Request
Send Confirmation Close
Close Connection
Close Communication

Figure 2. A schematic representation of the actions necessary to establish a connection between a client and a server, to exchange information and, finally, to close the connection. The General Listen Socket is a service-specific socket that is used by all clients to order a connection. The actual connection goes through a new pair of sockets.

2.2.3 Messages

Messages are defined according to a standardized language. This language mainly contains ASCII tokens, except for fields that contain binary data. These ASCII messages can easily be exchanged with many services on many hardware platforms without causing problems. Message fields can be presented in any order. The message language can be extended with new keywords by any application. In fact, each application is free in specifying extra keywords. The only restriction is that five fields need to be present:

(1) *message*. A field specifying the name of the function or message.

(2) *user*. The name of the user issuing the function.

(3) *from*. The name of the component that composed the message.

(4) *workstation*. A unique identification of the display the user is working from.

(5) *message id*. A unique string that can be used for reply messages and confirmations.

The fields valid in a message can be divided into two classes: optional and mandatory message fields. Mandatory fields have to be completed when the message arrives at the server, optional fields can be specified but are not necessary. The definition of the message language is shown in Figure 3.

```
request := (message)+
message := "BEGIN" value-part "END"
value-part := (keyword ":" value)+
value := list-of-values | raw-data | structure | single-value
list-of-values := "(" single-value ("," single-value)* ")"
raw-data := "<" NUMERIC ">" #bytes
structure := "{" value-part "}"
single-value := STRING | "STRING" | NUMERIC
```

Figure 3. Syntactical definition of the message language as used by HERMES. An asterisk indicates zero or more occurrences, and a plus means at least one occurrence. The raw-data value type is followed by exactly the number of bytes specified by the predecessing numeric. STRING and NUMERIC can be replaced by their ordinary definitions.

Each server has a message interpreter that reads the messages and checks them against the syntactical rules. These interpreters were developed with the yacc and lex tools of UNIX [15].

The macro recorder to define a series of instructions as one major command, consists of a tool that writes a series of messages to a file. This named file can be executed at any time, resulting in sending the messages to the corresponding services.

2.3 Kernel services

In the subsequent paragraphs, the different services of the new architecture will be outlined. The services are divided into two groups: the kernel services and the application services. The kernel services are essential for the client-server architecture and basically act as the operating system for the integration architecture. The application services provide the system with functionality and are not essential for the overall system's operation.

2.3.1 Message router

As already discussed in the Introduction, messages are not directly sent from a client to a server. There are a number of reasons to introduce a so-called message router, i.e., a server that accepts all messages and forwards them to the appropriate destination server.

Firstly, to make the system flexible and dynamically changeable, the link between a particular function or message and the server that can resolve it, is not stored in the individual components but resides in the router's database. This enables to influence the behavior of the system by changing the database. Moreover, new components can be added by defining a link between the message and the new component. New requests for that particular function can then be directed to the new component.

Secondly, the router can complement a message with its missing essential message fields. In this way, new components with new essential fields can still be dealt with. In addition, the message composition at the client's side can be limited to only those fields that are of interest. For each message field, the router database contains a default value.

Thirdly, the router records per service a preferred host and a list of alternative hosts. If a host does not respond to a connection request, alternative hosts can be tried. The router database contains per service a field that specifies whether the user should be prompted for confirmation of an alternative host. The same mechanism is used for messages. Each message contains a preferred server and a list of alternative servers. Upon failure to access the preferred server, the message router can decide to try an alternative server.

Fourthly, the router can be used to implement a broadcast mode. In this mode, the message is forwarded to all services on all hosts. Any service interested in the message can use it and perform the necessary actions. None of the components is aware of all the services that are available in the network. Therefore, the only component that can perform a broadcast is the router. With each message, a flag tells whether it is a broadcast message or a selective message.

Other aspects stored in the message router database are whether a server can have several sessions running for a single user. This field forces messages to a single session. However, the typical socket connection does not allow for easy reconnection with other components.

In addition, the router contains per service the name of the input format of data. This format restricts file selection requests from the service to files of a particular format.

2.3.2 Resource manager

Each user gets a personal resource manager assigned when logging in on a workstation. The typical name of this resource manager is the name of the user's display [e.g., X11 convention, mwi:0.0, meaning screen 0 at the console (also indicated by a 0) of machine mwi]. The router forwards all messages first to the personal resource manager. The personal resource manager will then adjust the message to the user's personal settings. This includes completion of optional message fields, overruling the host, and service assignment of the router.

The resource manager allows applications to connect with open servers. This feature is essential for servers that are limited to one server per user. A request for this type of server will activate the personal research manager to search its connection list for an open connection. If found, the message is forwarded on the associated socket descriptor, otherwise a new connection is established. The maximum number of open connections is limited to the number of open files an application can have.

The resource manager maintains per user a list of all open connections with servers. The resource manager registers per connection how many clients refer to the connection and the socket descriptor in a connection count. Connection-close instructions decrement the connection count. If this count is zero, the resource manager will send a close instruction to the corresponding server. When the resource manager gets a close instruction, it will first close all open connections.

The personal resource manager can also be used to forward a broadcast message to all its open socket connections. This mechanism allows servers that use data from a database to be hot-linked; any change to the original data will be broadcasted to the resource managers and consequently to the servers. Servers that are using the data directly reflect the change in their presentation.

2.3.3 User-interface server

When the user starts a session, the user-interface server will provide an interface on the screen. This server reads the user's service-info file and creates a user-interface on the screen. This user-interface contains the pretty-names of services that can be addressed from within the user-interface. Each service can be started and stopped from within the user-interface. One typical server is the macro recorder that can be used to play a macro sequence of messages. The macros are stored as a file containing this sequence of messages.

Typically, for a user-interface server for clinical research, the names of the following services could be presented:

(1) *data collection.* A service through which a database can be defined and new data can be entered.

(2) *data selection.* After having collected the data, one wants to make extractions for analysis. The data selection service is explained below.

(3) *data inspection.* A service that allows the user to inspect data graphically.

(4) *data manipulation.* Often, data selections have to be manipulated and rearranged to make them suitable for data analysis.

(5) *data analysis.* A service through which, e.g., statistical applications can be activated. This service provides the user with fill-up forms.

(6) *data presentation.* A service that makes graphical presentations of the data for inclusion in scientific papers, or for preparing slides or overhead sheets.

(7) *macro service.* A service that allows the storage of macros.

The service-info file contains per service an entry that specifies whether the service should be automatically started. Usually, the file server, print server, and message server are automatically started.

2.3.4 File server

The file server is the service that maintains the user's files. Independent of the computer on which the user logs in, the file server for that user will always be started at the computer where the files reside. This can be obtained by having the host specified for the file server as a resource manager's default; any request for the file server is forced to the desired host.

The file server can run in two modes: desktop mode and selective mode. Initially, the user interface will start the file server in desktop mode. All user files are shown as icons on the screen, and upon selecting a file, a pop-up menu is shown with all services that accept the file (type). In addition, services may have a file selection option. This option will send a message to the file server to show all files of a particular type, i.e., in selective mode. In this mode, files can only be selected, but not forwarded to other services. Whenever a server sends a request for files to the file server, its mode will temporarily change to the selective mode. Selection of a file returns the file server to desktop mode.

The file server allows the creation of new folders and the interactive management of the user's files. Each file is represented by an icon, and the icon depends on the type of the file. The file server interrogates the router about the services that are applicable to a particular file type. From this information, the file server is able to create pop-up menus with the services that can accept the file.

Upon selection of a file, the file server will send a message to the requesting service together with the data. When selecting a server associated with a file in desktop mode, the file server will first send a start message to the selected server, and secondly transmit the data.

A second option of the file server is to save files. Each server can send a store message to the file server with the data to be saved. The file server will store the data in the user's directory and update his file-info base. This file-info base contains per file additional information, such as the type of the file, the expiration date, the name of the server that produced the file and, if the file was derived from other data, the name of the original file.

In addition to this file-info base, the file server also uses a type-info base, that describes the different file types. It contains per file type the name of the type, the icon associated, and the drivers to send and store the data. These drivers are activated by the file server whenever data have to be sent as a message or have to be translated from the message format into a data file. New types can be added by using a special option of the file server to edit its type-info base. For each new type, a store-and-send driver has to be created.

2.3.5 Data model server

In addition to having data in files, data can also be stored in a database management system. The data model server contains a description of the data attributes of all databases accessible from the workstation (see Figure 1a,b). The data model server contains a data dictionary and an organization of the data along medically interrelated concepts and instances [16]. Instances represent the actual variables stored in the databases, and concepts are used to define commonalities of the instances.

For coded variables, the data model contains all possible code values with their associated meaning. Relationships between instances can also be modelled in this object base. These relationships represent the medical relations that exist between the variables or the entities. For instance, a relation: "has abnormality or dysfunction" could link, e.g., the entity: "right ventricle" with the entity "Right Ventricular Hypertrophy".

The objects (concepts and instances) are stored in an object-oriented base that is accessible through the data model server. This data model server accepts request messages for information about data. An interactive graphical tool can be used to edit the objects.

2.3.6 Data selection server

New data files can be obtained by selecting data from the databases. A data selection server provides the user with an interface that allows the user to select data in a friendly way: the data selection server obtains from the data model server the organization of the data and displays them on the screen. Data can be retrieved by selecting either the individual variables (e.g., blood pressure) or aggregations of data (e.g., laboratory data). Variables and concepts are presented to the user with pretty names.

Specification of selection criteria is also provided by this server. The data model server provides the selection server with the type and the possible values of the variables. Code lists can be shown as flat files or as a hierarchically organized structure. Criteria that use the medical relationships among the codes can also be used. For instance, a question such as: "select the age and name of all patients that have an abnormality or dysfunction of the right ventricle" can be solved with information from the data model server.

After having completed the specification of the selection, the server will compose a message with an SQL+ query. This SQL+ query contains per variable an extended address. Standard SQL is limited to a relation-attribute tuple. SQL+ extends this notion to a quadruple of dbms-database-relation-attribute (see Figure 4). The selection clause is extended with expressions for joining records from different relations. This SQL+ message is forwarded to the multi-database server.

```
select
  ingres.andro.semen.volume, his.regist.andrology.diagnosis
from
  ingres.andro.semen, his.regist.andrology
where
  ingres.andro.semen.volume > 5.1 and
  his.regist.andrology.diagnosis" = "unknown diagnosis" and
  ingres.andro.semen.patid = his.regist.andrology.patientid
```

Figure 4. An example of an SQL+ script. The specification of attributes is extended with the name of the database management system, the database name, and the relation name. This enables to define SQL scripts that use data from different databases and database management systems.

2.3.7 Multi-database server

The multi-database server accepts messages containing an SQL+ query. The server expands an SQL+ query to individual standard SQL queries for the various database management systems involved. Selection criteria are analyzed by the server and, if possible, passed to the specific database server. Variables that are used in the expressions but not contained in the variable part of the SQL+ message are also required from the specific database servers. With all these variables available, the multi-database server is capable to evaluate conditions that involve data from various databases.

Subsequently, the SQL queries are passed to the specific database management servers. Data obtained from these servers are joined by the multi-database server to contain one data set and the conditions are evaluated on this dataset. Finally, only the data of the variables requested in the SQL+ message are returned to the caller.

2.3.8 Database service

Each database management system is encapsulated by a database service that controls the access to the database management system. This database service accepts standard SQL and translates it to the database-specific query language. Results from this query are returned to the caller. When the database management system does not provide an evaluation of conditions, the database service has to take care of it. In that case, variables referred to in the condition should also be retrieved from the database management system. Obvious examples of database services are those for INGRES, ORACLE, and dBaseIII. In addition, the power of our approach is shown by a service under development which allows similarly uniform access to databases residing on a Hospital Information System. In that project, links to both a database pertaining to the Department of Andrology and a (part of) the database of the Laboratory for Clinical Chemistry are developed.

2.3.9 Presentation server

Many application services require a graphical presentation for their data. To avoid implementation of this functionality in each application service, a presentation service has been added to the workstation environment. This server accepts messages containing data with a request for a particular presentation format: histogram, table, pie-chart, line-draw, etc. With this service, manipulation functions for the presentations are directly available. The presentation server also contains facilities to export the picture to other (commercial) packages such as Harvard Graphics or SlideWrite.

2.3.10 Print server

All print operations of any service are not directly sent to the printer, but passed on to the print service. This server will display a printer icon on the user's desktop while busy. This icon can be opened to give more information about the print queue and the status of the print job.

2.3.11 Help server

Requests for help information are not solved individually by applications, but are passed on to a help service. This help service provides the user with a hypertext interface; a help-text block contains words that are marked. Selection of these marked words gives the user access to detailed information about that subject. This service allows the user to go beyond the scope of the help information about one service and discover the relation between the services. New services are obliged to provide their own help files according to a specified format.

2.3.12 Message service

Any error occurring in a service is reported to the message server. This server will use the user's language to report the error on the display. Besides errors, this service can also be used to display any information from the applications. The message service uses catalog files to translate an internal message number to textual information. The message service provides the user with options to inspect a logfile of messages according to a specific filter.

2.4 Application services

The workstation's integration architecture allows new applications to be incorporated very easily: the router base has to be updated with information about the new application service, and a help file has to be stored with the help service. New data types can be defined by updating the file server's typeinfo base.

Often, data obtained from database management systems have to be manipulated before they can be used for statistical tests. A *data manipulation service* provides the user with an interactive tool for manipulating the data. This includes restriction of a Cartesian product to time-interval criteria (e.g., not all combinations of records of relation A containing a laboratory test, with all records of relation B containing information about a visit are relevant: there is a certain time interval that relates lab data with a visit). Other manipulation functions are max, min, avg, sum, etc. The manipulation service allows the user to define his own manipulations as a front-end to the UNIX awk interpreter.

The *statistical application server* of the workstation runs as an encapsulated service for BMDP. The server accepts requests for statistical operations and returns analysis results in a standardized format. The server translates the incoming message to a valid BMDP command script and transforms the associated data to the specific format of BMDP. The server hides the typical structure of BMDP from the workstation, allowing BMDP to be replaced by any equivalent statistical package, such as SPSS or SAS. Other statistical tools include a service that encapsulates the ISPAHAN *statistical pattern recognition* program [17].

For the medical domain, typical services such as an *X-ray image server* or an *ECG interpretation server* can be added that can be applied to typical data files. The image server provides the user with a tool to manipulate images (e.g., 3D reconstruction or video) that is resident to dedicated image analysis hardware. The ECG server allows the user to automatically obtain an interpretation of an ECG [18] and to inspect the signal graphically.

2.4.1 Protection and security

The architecture philosophy of the workstation is to provide an open systems environment. Each new application can use the existing services for typical functions (new data, open file, save file, print, make graph, help, etc.). However, this open environment should not deny the security and protection schemes necessary to protect data and software against illegal use. Therefore, a number of protection and security rules have been defined that can be distinguished at the following levels.

The first level of security is the UNIX login mechanism. Each user has to login and provide the system with a secret password. This is checked against information stored in an encrypted file. The second level is provided by the Internet daemon. This daemon extracts messages from the network and passes them on to the required services. The daemon inspects a security file to see whether the computer where the message originates from is allowed to address the service at that time. The protection can be specified by the system manager in the daemon's database. The third level of protection can be forced by the service itself. Any start request can be checked by the service for authority. The service can also prompt the user for a specific password (e.g., the database service may do so).

2.5 Standards

The workstation has been developed as much as possible along presently available industrial standards. This standardization concentrates on two aspects: (1) portability of services to different platforms and (2) communication between services.

The major developments follow the UNIX System V standard and are programmed in ANSI C. The services use Berkeley sockets to communicate with each other through a TCP/IP Ethernet protocol. Graphical user-interfaces are developed with the X11 OSF/Motif environment. Messages are mainly ASCII formatted. Database queries are in the messages expressed as ANSI SQL.

2.6 Tools

In order to increase the standardization of the services and to ease their development, a number of tools can be used: the Lex and Yacc tool [18] provide the message interpreter for each service. To stimulate encapsulation of existing applications, tools are available that give high-level support for the creation of a data translator and a command generator. A front-end to the UNIX facility, awk [19], allows specification of a data translation facility. A macro recorder in combination with the stream editor sed [20] supports the generation of a command generator.

Libraries are available to send and accept messages and to easily access object-oriented databases.

Graphical user-interfaces can be constructed with a graphical user-interface management system; the constructed user-interface can be written as C-code and be used in all kinds of applications. Hewlett Packard's Encapsulator is used to encapsulate the access to existing applications. A source-code management system together with an integrated debugging environment is also provided.

2.7 Integration of PCs

The availability of X-servers for Windows 3.0 allow the integration of PCs in this architecture. These PCs can either access a user-interface service developed for OSF/Motif or can have their own Windows 3.0 user-interface service. This Windows 3.0 service addresses through the network the other services which can provide their interactive output through the PC X-server on the user's desktop.

3 Conclusion

The maturing of network and user-interface technology provide us with environments that can link computers of various vendors in one network. The integration architecture outlined in this paper extends this notion of communication to the application or service level. The architecture is developed around a message router and personal resource manager that allow new applications to be added dynamically. In addition, these two services reduce the inter-dependencies of the services, thus limiting the complexity of managing these environments.

The message format has been chosen to be suited for addressing functions in other services. Typical standards such as HL7, Medix or Edifact still concentrate on data communication and do not provide facilities for addressing functions [21]. However, the architecture described in this article could be extended easily with services that translate these standards into the workstation format and vice versa. In addition, bridges can be defined to exchange messages with other client-server environments such as those of Helios [9] and Hewlett Packard's Physician's Workstation [8].

Cooperation should be established with other groups working in the same direction to standardize message formats and message contents; for data operations, SQL could be selected as a standard, for statistical and graphical operations no standards are as yet available. Research should be directed towards how nomenclature and terminology could be standardized, for instance according to ICD-9 or 10 or UMLS.

Future research, based on these open system environments, can be directed towards other task domains, such as direct patient care, quality assessment, and education. Typically, research should focus on user modelling and desktop metaphors that are best suited for clinicians. Before establishing such an open system environment for supporting professional tasks, the following problems have to be addressed:

1. Vendors of medical information systems and equipment should be required to make their systems open, i.e. accessible by other applications.
2. Standards should be defined and used for exchange of medical data, for addressing processing procedures, for medical terminology and for network connectivity.
3. A number of vendors and groups should participate in order to provide within a reasonable time an open system environment with a critical mass of functionality.

4. The open approach requires systems to be more spread around several workstations instead of on one central system; this decentralized organization of hardware has an immediate impact of support and maintenance.
5. In the meantime, a first step to having a more open system environment is the encapsulation of existing applications, as suggested in this paper.

Acknowledgements
We gratefully acknowledge the contribution of R.F. Weber (Dept. of Andrology) and of A.J. van Ooijen, (CDAI), both of the University Hospital Dijkzigt Rotterdam, to the database service for accessing Andrology data in the Hospital Information System.

References

[1] Van Mulligen EM, Timmers T, Leao B de F. Implementation of a medical workstation for research support in Cardiology. In: Miller, R., eds. *Proceedings of the 14th Symposium on Computer Applications in Medical Care.* New York: IEEE Computer Society Press, 1990: 769-73.

[2] Bowen D. Open distributed processing. Computer Networks and ISDN Systems. 1991; *23*: 195-201.

[3] Greenes RA. Promoting productivity by propagating the practice of "plug-compatible" programming. In: Miller, R., eds. *Proceedings of the 14th Symposium on Computer Applications in Medical Care.* New York: IEEE Computer Society Press, 1990: 22-6.

[4] Meijler TD, Kuil HW, Gelsema ES, Kersten ML. Bridging the boundaries between applications: providing interactive interoperability for the end-user. In: Kambayashi Y., Rusinkiewicz M., Sheth A., eds. *Proceedings of the First International Workshop on Interoperability in Multidatabase Systems.* Los Alamitos: IEEE Computer Society Press, 1991: 338-41.

[5] Van Mulligen EM, Timmers T, Van den Heuvel F. A framework for uniform access to data, software and knowledge. In: Clayton, P.D., eds. *Proceedings of the 15th Annual Symposium on Computer Applications in Medical Care.* New York: McGraw-Hill, 1991: 496-500.

[6] Lunin LF, Ball MJ. Perspectives on Integrated Academic Information Systems (IAIMS). Amer Soc Inform Science 1988; *39*: 102-12.

[7] Tang M, Annevelink J, Fafchamps D, Stanton WM, Young C. Physicians' workstations: integrated information management for clinicians. In: Clayton, P.D., eds. *Proceedings of the 15th Annual Symposium on Computer Applications in Medical Care.* New York: McGraw-Hill, 1991: 569-73.

[8] Degoulet P, Coignard FCJ, Laurent MC, et al. The HELIOS European project on software engineering in medical informatics. In: Timmers T, Blum BI, eds. *Proceedings of the IMIA Working Conference on Software Engineering in Medical Informatics.* Amsterdam: Elsevier Science Publ, 1990: 125-37.

[9] Barsalou T. View objects for relational databases. PhD thesis, Medical Information Sciences Program, Stanford University, 1990.

[10] Greenes RA. Desktop Knowledge. A new focus for medical education and decision support. Methods Inf Med. 1989; *28*: 332-9.

[11] Reiss SP. Connecting tools using message passing in the Field environment. IEEE Software, 1990; *3*: 57-66.

[12] Chin R, Chanson ST. Distributed object-based programming systems. ACM Computing Surveys. 1991; *23*; 91-124.

[13] Robinson D. Remote procedure call: a stepping stone towards ODP. Computer Networks and ISDN Systems. 1991; *23*: 191-4.

[14] Zimmermann H. OSI reference model - the ISO model of architecture for open systems interconnection. IEEE Trans Commun 1980; *28*: 425-32.

[15] Lesk ME. Lex - A lexical analyzer generator. Comp Sci Tech Rep 39. Murray Hill, NJ: Bell Laboratories, 1975.

[16] Timmers T, Van Mulligen EM, Van den Heuvel F. Integrating clinical databases in a medical workstation using knowledge-based modelling. In: Lun K.C., Degoulet P., Piemme T.E., Rienhoff O., eds. Proceedings of the 7th World Congress on Medical Informatics. Amsterdam, North-Holland Publ, 1992: 478-82.

[17] Gelsema ES. ISPAHAN; an interactive system for pattern analysis; structure and capabilities. In: Gelsema ES, Kanal LN, eds. Pattern Recognition in Practice. Amsterdam: North-Holland Publ, 1980: 481-91.

[18] Van Bemmel JH, Kors JA, Van Herpen G. Methodology of the Modular ECG Analysis System MEANS. Methods Inf Med. 1990; *29*: 346-53.

[19] Aho AV, Kernighan BW, Weinberger PJ. The AWK Programming Language. NY: Addison-Wesley, 1988.

[20] Hewlett Packard Company. Text Processing: User's Guide. 1991.

[21] Ostler DV, Harrington JJ, Hannemyr G. A common reference model for healthcare data exchange - P1157 medix system architecture. In: Miller, R., eds. *Proceedings of the 14th Annual Symposium on Computer Applications in Medical Care.* Los Alamitos: IEEE Computer Society Press, 1990: 235-8.

CHAPTER 8

Summary and Conclusions

Summary

This thesis addresses two main questions: (1) How to build a flexible and extensible architecture for a medical workstation that integrates in a network existing applications that are not to be modified, and (2) how to support clinical users by systems integration in a network for biomedical research and patient care.

The first part of this thesis describes the development, design and evaluation of a prototype integrated medical workstation, for the support of clinical data analysis. The second part presents a new architecture for an integrated medical workstation.

In **Chapter 1**, the Introduction, the issues related to the development of an integrated medical workstation are outlined: decentralization of hardware, network technology, user interface systems, and standardization efforts. In this chapter, the various approaches to integration, such as post-facto, pre-facto, data integration, functional integration and semantic integration are presented.

The prototype integrated medical workstation

In **Chapter 2**, the requirements for a prototype integrated medical workstation for the support of clinical data analysis are specified. In addition, the design and the implementation of this prototype are outlined. The technical possibilities of graphical user interfaces, network technology and the client-server model for the development of an integrated medical workstation are investigated, and its limitations are discussed.

Clinicians are more and more confronted with a wide variety of databases and programs. Often, there are large differences between the commands the databases use for retrieving data and the commands that can be used to start programs. In **Chapter 3**, the databases and applications integrated in the prototype workstation are presented. Attention is paid to the central data model that is used for multi-database data retrieval and data entry.

Evaluation of the prototype integrated medical workstation

In **Chapter 4**, the results are presented of the user evaluation of an integrated medical workstation for support of clinical research. Twenty-seven users were recruited from medical and scientific staff of the University Hospital Dijkzigt, the Faculty of Medicine of the Erasmus University Rotterdam, and from medical institutions outside the Erasmus University.

For instruction of the users, a written, self-contained tutorial was given. Subsequently, an experiment was done in which six clinical data analysis problems had to be solved, and an evaluation form was filled out. The aim of this user evaluation was to obtain insight in the benefits of integration for support of clinical data analysis for clinicians and biomedical researchers. The problems were divided into two sets, with gradually more complex problems. In the first set users were guided in a stepwise fashion to solve the problems. In the second set each stepwise problem had an open counterpart.
During the evaluation, the workstation continuously recorded the user's actions and these recordings were used as raw data for the evaluation. Significant differences became apparent between clinicians and non-clinicians for the correctness (means 54% and 81%, respectively, p=0.04), completeness (means 64% and 88%, respectively, p=0.01), and number of problems solved (means 67% and 90%, respectively, p=0.01). These differences were absent for the stepwise problems.

Clinicians tend to skip more problems than biomedical researchers. No statistically significant differences were found between users with and without clinical data analysis experience, for correctness (means 74% and 72%, respectively, p=0.95), and completeness (means 82% and 79%, respectively, p=0.40). Our main conclusion was that the performance of a clinician was raised to the level of someone experienced in clinical data analysis and that various clinical research problems can be solved easily with support of the workstation.

The results of this experiment have been used for the development of a general architecture for the successor of this prototype workstation. From the assessment study, it appears that the evaluation of a system such as the prototype workstation with its high user-interaction is very complex. We also realized that references in the literature on quantitative rather than qualitative assessments are very sparse if not totally absent. Yet, we were convinced, and still are, that progress in the further development of these and similar systems can only be made when the evaluation of such systems is done in a quantitative and objective as possible manner. The evaluation of our prototype integrated workstation serves as a reference model for the assessment of future developments in this domain. Getting insight in the type of errors users make, may help in developing new user-interfaces, desktop metaphors, and user support modules. It allows to repeat this evaluation for new developments and to compare the outcome with the current results.

HERMES: a generalized architecture for an integrated medical workstation
An object-based software integration architecture is described in **Chapter 5** which provides communication within a network between applications according to the client-server model. Each object in this object-based architecture is called an accessor and hides the details for accessing data and functions from other applications.

This architecture has been designed to integrate existing applications. A request for data and functions directed to the accessor framework is served by an accessor server that analyzes the request, completes it with essential parameters, and forwards it to the application service object. This application service object hides the access to the data and functions available in an application. For existing applications, this application service object is served by a separate application service process that converts the accessor requests into instructions for the application.

Newly developed applications can directly accept requests to the corresponding application service object. A sequence of accessor requests can be stored as a named program and processed by the accessor server upon request. The accessor object-base can be edited so as to contain the most recent information about the available data and function. The accessor framework has been made operational in a network with servers and workstations.

Medical workstations are increasingly introduced as the integration center of various information streams. In **Chapter 6**, the intrinsic features for integration, such as connectivity, modularization, shareability, abstraction and encapsulation, are described together with their impact on an integration architecture for a medical workstation for existing applications. Using these features at the onset, a new vision on a medical workstation is outlined. An example of a new architecture for such an integrated medical workstation is sketched.

In **Chapter 7**, an architecture is described that integrates existing applications in a network-wide system. The architecture follows the new open software paradigm, and defines kernel and application services that collaboratively solve the tasks of end-users and provide them with an intuitive user interface. This chapter describes the message language and the kernel mechanism for addressing application services.

Concluding remarks

From the results presented in this thesis, it is clear that an integrated medical workstation can be developed on top of existing applications, without the need to modify them.

This integration frees users from having to know how to connect through the network, how to exchange data between applications and how to operate those applications. The user interface of the workstation hides all the inter-application differences and provides users with a medically-oriented interface.

The user evaluation showed that clinicians can benefit from such an integrated approach and are able to solve most of six representative clinical data analysis problems within about one and a half hour with the integrated workstation after a short, self-contained tutorial of two hours.

Users with no clinical data analysis experience solved the problems with a performance comparable to those with experience.

The refinement of the architecture according to what has been found during the user evaluation and from the development of the prototype has led to a generalized architecture for an integrated medical workstation. This generalized architecture can be easily extended with new applications without any consequence for the applications already integrated.

Given the increasing number of medical informatics applications in the clinical environment, both as industrial products and as products of matured research, there is an increasing problem of how to combine these different applications and how to make them accessible to the clinician. We believe to have shown that our workstation architecture provides a flexible and extensible solution to the needed advanced level of integration.

HOOFDSTUK 8

Samenvatting en Conclusies

Samenvatting

Dit proefschrift richt zich op twee vragen: (1) Hoe kan een flexibele en uitbreidbare architectuur voor een medisch werkstation gemaakt worden zodat bestaande applicaties zonder aanpassingen in een netwerk geïntegreerd kunnen worden, en (2) hoe kunnen clinische gebruikers door systeem integratie ondersteund worden bij hun clinisch wetenschappelijk onderzoek en hun patiëntenzorg.

Het eerste deel van dit proefschrift beschrijft de ontwikkeling, het ontwerp en de evaluatie van een prototype medisch werkstation ter ondersteuning van clinische gegevens analyse. Het tweede deel beschrijft een nieuwe architectuur voor een geïntegreerd medisch werkstation.

In **Hoofdstuk 1**, de inleiding, worden de aspecten die gerelateerd zijn aan de ontwikkeling van een medisch werkstation beschreven: decentralisatie van hardware, netwerk technologie, gebruikers-interface systemen en het streven naar standaardisatie. In dit hoofdstuk worden ook de verschillende benaderingen van integratie uiteengezet, zoals post-facto en pre-facto integratie, functionele integratie en semantische integratie.

Het prototype geïntegreerd medisch werkstation

In **Hoofdstuk 2**, worden de voorwaarden voor een prototype geïntegreerd werkstation ter ondersteuning van clinische gegevens analyse gespecificeerd. Daarnaast worden het ontwerp en de implementatie van dit prototype beschreven. De technische mogelijkheden van grafische gebruikers-interfaces, netwerk technologie en het client-server model voor de ontwikkeling van een geïntegreerd medisch werkstation worden onderzocht, en de beperkingen ervan worden besproken.

Clinici worden in toenemende mate geconfronteerd met een grote verscheidenheid van gegevensbestanden en applicaties. Er zijn dikwijls grote verschillen tussen de commando's die gegevensbestanden gebruiken voor het ophalen van gegevens en tussen de commando's waarmee applicaties kunnen worden bediend. In **Hoofdstuk 3** worden de gegevensbestanden en applicaties beschreven die in het prototype medisch werkstation geïntegreerd zijn. Er wordt aandacht gegeven aan het centrale data model dat gebruikt wordt voor het ophalen en invoeren van gegevens in verschillende gegevensbestanden.

Evaluatie van het prototype geïntegreerd medisch werkstation.
In **Hoofdstuk** 4 worden de resultaten van een gebruikers-evaluatie van het geïntegreerde medische werkstation ter ondersteuning van clinische gegevens analyse beschreven. Zeven-en-twintig gebruikers van de medische en wetenschappelijke staf van het Academisch Ziekenhuis Rotterdam, de faculteit der Geneeskunde en Gezondheidswetenschappen en van medische instellingen buiten deze werden gevraagd om aan de evaluatie mee te doen.

De gebruikers volgden een geschreven zelf-instructie cursus. Hierna werd een experiment uitgevoerd waarin zes clinische gegevens analyse problemen opgelost moesten worden, en een evaluatie formulier ingevuld moest worden. Het doel van deze gebruikers-evaluatie was het verkrijgen van inzicht in de voordelen van integratie voor de ondersteuning van clinische gegevens analyse van clinici en biomedische onderzoekers. De problemen werden in twee groepen ingedeeld, waarbij de complexiteit van de problemen toenam. Bij problemen uit de eerste groep werden de gebruikers stapsgewijs geleid naar de oplossing. Voor elke stapsgewijs probleem was er een vergelijkbaar open probleem in de tweede groep.

Tijdens de evaluatie registreerde het werkstation voortdurend de acties van de gebruiker en deze registraties werden als ruwe gegevens voor de evaluatie gebruikt. Tussen clinici en niet-clinici werden significante verschillen gevonden bij de open problemen voor wat betreft de correctheid (gemiddelden van respectievelijk 54% en 81%, p=0.04) en de compleetheid (gemiddelden van respectievelijk 64% en 88%, p=0.01) en het aantal opgeloste problemen (gemiddelden van respectievelijk 67% en 90%, p=0.01). Deze verschillen werden niet gevonden voor de stapsgewijze problemen.

In het algemeen sloegen clinici meer problemen over dan biomedische onderzoekers. Er konden geen significante verschillen tussen gebruikers met en zonder clinische gegevens analyse ervaring voor wat betreft de correctheid (gemiddelden van respectievelijk 74% en 72%, p=0.95), en de compleetheid (gemiddelden van respectievelijk 82% en 79%, p=0.40) vastgesteld worden. Onze hoofd-conclusie was dat de prestatie van een clinicus verhoogd werd tot die van iemand die ervaring heeft met clinische gegevens analyse en dat verschillende clinische onderzoeks-problemen gemakkelijk met het werkstation opgelost kunnen worden.

De resultaten van dit experiment zijn gebruikt voor de ontwikkeling van een opvolger van het prototype werkstation. Het blijkt uit deze studie dat de evaluatie van zo'n prototype werkstation met een sterke gebruikers-interactie ingewikkeld is. Het was ook duidelijk dat er nauwelijks of geen literatuur is die een kwantitatieve in plaats van een kwalitatieve evaluatie beschrijft. Toch waren we overtuigd, en dat zijn we nog steeds, dat er alleen maar vooruitgang geboekt kan worden bij de ontwikkeling van zulke systemen als de evaluatie ervan op een zo kwantitatieve en objectieve mogelijke manier gebeurt. De evaluatie van ons prototype geïntegreerd werkstation dient als een referentie model voor de evaluatie van toekomstige ontwikkelingen op dit gebied. Het verkrijgen van inzicht in de soorten fouten die gebruikers maken, kan bijdragen tot de ontwikkeling van nieuwe gebruikers-interfaces, beeldscherm-metaforen en gebruiker-ondersteunende modules. Het stelt ons in staat deze evaluatie te herhalen voor nieuwe ontwikkelingen en de resultaten ervan te vergelijken met de huidige.

HERMES: een gegeneraliseerde architectuur voor een geïntegreerd medisch werkstation
In **Hoofdstuk 5** wordt een object-gebaseerde software integratie architectuur beschreven die de communicatie tussen verschillende applicaties in een netwerk verzorgt volgens het client-server model. Elk object in deze object-gebaseerde architectuur wordt een *accessor* genoemd en verbergt de details om gegevens en functies te benaderen voor andere applicaties.

Deze architectuur is ontwikkeld om bestaande applicaties te integreren. Een aanvraag voor gegevens en functies wordt gericht tot het *accessor systeem* dat beheerd wordt door een *accessor server* die de aanvraag analyseert, essentiële parameters toevoegd en het doorstuurt naar een applicatie service object. Zo'n applicatie service object verbergt de toegang tot gegevens en functies die beschikbaar zijn in een applicatie. Bestaande applicaties worden bediend door een apart service proces dat de aanvraag van de accessor vertaalt naar instructies voor de applicatie.

Nieuw ontwikkelde applicaties kunnen direct reageren op aanvragen van het applicatie service object. Een aantal accessor aanvragen kan worden opgeslagen als een programma dat, wanneer nodig, verwerkt kan worden door de accessor server. Het accessor systeem is operationeel gemaakt in een netwerk met servers en werkstations.

In toenemende mate worden medische werkstations geïntroduceerd als het integratie middelpunt van verschillende informatie bronnen. In **Hoofdstuk 6** worden de intrinsieke eigenschappen voor integratie, zoals connectiviteit, modularisatie, gemeenschappelijk gebruik, abstractie en encapsulatie beschreven met hun effect op de integratie architectuur van een medisch werkstation. Uitgaande van deze eigenschappen wordt een nieuwe visie op een medisch werkstation beschreven. Een voorbeeld van een nieuwe architectuur voor zulke geïntegreerde werkstations wordt geschetst.

In **Hoofdstuk 7** wordt een architectuur beschreven waarmee bestaande applicaties in een netwerk geïntegreerd worden. De architectuur volgt het open software paradigma en definieert kernel en applicatie services die gezamenlijk eind-gebruiker taken oplossen, waarbij deze services met een intuïtief gebruikers-interface zijn uitgerust. Dit hoofdstuk beschrijft de berichten-taal en het kernel mechanisme voor het benaderen van de applicatie services.

Afsluitende opmerkingen
Uit de resultaten zoals ze in dit proefschrift gepresenteerd worden, komt duidelijk naar voren dat het mogelijk is om een geïntegreerd werkstation te ontwikkelen uitgaande van bestaande applicaties, zonder dat deze aangepast hoeven te worden.

Deze integratie vermindert de noodzaak voor gebruikers om te weten hoe men via het netwerk verbindingen kan maken, hoe gegevens tussen applicaties uitgewisseld kunnen worden en hoe de verschillende applicaties aangeroepen dienen te worden. De gebruikers-interface van het werkstation verbergt alle verschillen die er tussen applicaties bestaan en biedt de gebruiker een medisch-georienteerd interface.

De gebruikers-evaluatie toont aan dat clinici geholpen worden door een dergelijke geïntegreerde aanpak en dat ze in staat zijn de meeste van de zes representatieve clinische gegevens analyse problemen binnen ongeveer anderhalf uur op te lossen met het geïntegreerd medisch werkstation, dat zij na een korte zelf-instructie van ongeveer twee uur te hebben gedaan.

Gebruikers zònder ervaring met clinische gegevens analyse losten de problemen op met een resultaat vergelijkbaar met dat van personen met ervaring.

De verfijning van de architectuur ten gevolge van wat uit de gebruikers-evaluatie naar voren kwam, heeft geleid tot een generalisatie van de architectuur voor een geïntegreerd medisch werkstation. Deze gegeneraliseerde architectuur kan gemakkelijk uitgebreid worden met nieuwe applicaties zonder enig gevolg voor de applicaties die al geïntegreerd zijn.

Gegeven het toenemend aantal medisch informatica applicaties in de clinische omgeving (zowel industriële producten als producten die het resultaat zijn van uitgekristalliseerd onderzoek), wordt het probleem steeds nijpender, hoe deze verschillende applicaties kunnen worden gecombineerd en voor de clinicus toegankelijk gemaakt. Wij denken aangetoond te hebben dat onze werkstation architectuur een flexibele en uitbreidbare oplossing biedt aan het hiertoe noodzakelijke geavanceerde niveau van integratie.

APPENDIX A

Original tables containing evaluation data

Table 1. Raw data, as collected during the assessment and as obtained from an evaluation form and MedLine.

Uid	Cl	Md	T_c	T_e	I	II_a	II_b	III	IV_a	IV_b	V_a	V_b	VI	Cld	Sta	Cp	Tsa	CI_1	CI_n	P_1	P_n
1	+	+	53	10.2	4	4	4	4	4	4	4	4	4	3	3	3	3	0	33	1	9
2	+	+	156	12.9	4	4	4	4	4	4	4	1	4	3	3	3	3	104	399	20	81
3	+	+	145	9.0	4	4	4	4	4	4	3	1	4	4	3	3	2	20	99	5	19
4	+	+	140	8.4	4	4	4	4	4	4	3	-	-	4	4	3	2	51	76	8	14
5	+	+	150	8.0	3	3	1	4	4	1	3	-	-	4	4	3	2	30	53	5	13
6	+	+	120	14.0	3	2	-	-	-	-	-	-	-	4	2	3	2	26	82	6	19
7	+	+	179	11.7	4	4	4	4	4	4	3	-	-	3	2	3	2	0	7	0	3
8	+	+	141	29.5	4	2	-	-	-	-	-	-	-	3	2	3	2	0	2	0	1
9	-	-	144	9.0	4	4	4	4	4	4	4	4	4	1	4	2	1	22	22	3	4
10	-	-	170	8.4	4	4	4	4	4	4	4	4	4	4	4	2	3	7	117	1	23
11	-	-	155	11.9	4	4	4	4	4	4	4	1	4	2	4	3	2	24	53	5	14
12	-	+	101	12.4	4	4	3	4	4	4	4	1	4	4	4	2	2	59	199	13	43
13	-	+	67	6.5	4	4	4	4	4	4	4	4	4	1	3	3	2	30	45	6	10
14	-	+	113	7.5	4	4	4	4	4	1	4	4	4	3	3	3	3	29	56	4	11
15	-	-	106	10.4	4	4	1	4	4	3	3	1	4	4	5	2	1	0	306	0	56
16	-	-	103	12.0	3	4	4	4	4	4	4	1	4	3	3	3	3	0	0	0	1
17	-	+	117	15.5	3	2	3	2	-	-	-	-	-	4	4	3	3	0	9	0	1
18	-	-	120	6.3	4	4	4	4	4	4	4	4	4	4	4	2	2	0	0	0	0
19	-	-	93	6.7	4	4	4	4	4	4	4	4	4	3	3	3	3	0	0	0	0
20	-	-	104	8.4	4	4	4	4	4	4	4	4	4	2	4	2	2	0	0	0	0
21	-	-	81	6.8	4	4	3	4	4	4	4	4	4	1	2	2	2	0	0	0	0
22	-	-	157	8.6	4	3	4	4	4	4	4	4	4	4	4	1	1	0	0	0	0
23	-	+	88	9.3	4	4	4	4	4	4	4	4	4	2	2	3	2	0	0	0	0
24	-	-	161	6.7	4	3	4	4	4	4	4	4	4	2	3	3	3	0	0	0	0
25	-	-	108	11.8	4	4	3	4	4	4	4	4	4	3	2	3	3	0	0	0	0
26	-	+	60	30.7	4	3	1	4	-	-	-	-	-	3	3	3	3	0	0	0	0
27	-	-	154	-	-	-	-	-	-	-	-	-	-	1	2	2	3	0	0	0	0

Uid	= user identification number,
Cl	= clincian, directly involved in patient care,
Md	= Medical doctor,
T_c	= time course (min.),
T_e	= average time per question answered (min.),
I-VI	= status per question (4=correct completed,3=incorrect completed,2=incorrect and incomplete,1=not done,-=quitted) ($_a$ and $_b$ parts are equivalent questions for different patient selections),
Cld	= clinical data analysis experience (5=expert,4=much,3=normal,2=little,1=none),
Sta	= statistical experience (5=expert,4=much,3=normal,2=little,1=none),
Cp	= subjective comparison with current support for clinical data analysis (3=better,2=equal,1=worse),
Tsa	= subjective estimation of time savings of workstation approach (3=better,2=equal,1=worse),
CI_1	= citation index summed over MedLine 1987-1992, if first author,
CI_n	= citation index summed over MedLine 1987-1992, if author,
P_1	= number of publications MedLine 1987-1992, if first author,
P_n	= number of publications MedLine 1987-1992, if author.

Table 2. Summary table for results derived from the initial raw data.

	#	T_c	T_e	S_1	S_2	S_3	$S_{1\alpha}$	$S_{1\beta}$	S_4	C	CI_1	CI_n	P_1	P_n	CI_1/P_1	CI_n/P_n
Cl	8	136	13.0	0.54	0.64	0.67	0.69	0.43	0.33	0.69	28.9	93.9	5.6	21.1	3.2	4.0
NCl	18	114	10.5	0.81	0.88	0.90	0.82	0.81	0.10	0.87	9.5	44.8	1.8	9.1	2.0	2.4
E	13	125	9.9	0.74	0.82	0.83	0.82	0.68	0.17	0.83	30.9	118.5	5.9	25	4.8	4.8
NE	13	116	12.6	0.72	0.79	0.82	0.73	0.71	0.18	0.80	0	1.4	0	0.46	-	1.0
SE	11	133.1	9.8	0.73	0.83	0.85	0.77	0.70	0.15	0.80	17.5	75.9	3.2	15.27	0.50	0.45
NSE	15	111.2	12.4	0.73	0.79	0.81	0.78	0.69	0.19	0.83	13.9	48.2	2.8	10.27	0.31	0.31
Md	14	116	13.3	0.58	0.67	0.71	0.70	0.50	0.29	0.72	24.9	75.7	4.9	16	3.1	3.9
NMd	12	125	8.9	0.90	0.96	0.96	0.88	0.92	0.04	0.93	4.4	41.5	0.8	8.2	1.6	1.7

Cl/NCl	= involved in patient care/not involved
E/NE	= experience of data analysis derived from publications MedLine/no experience
SE/NSE	= statistical experience as stated in the evaluation form/no experience
Md/NMd	= having a medical degree/no medical degree
#	= number of users
T_c	= time of course (min.)
T_e	= average time per question answered (min.)
S_1	= fraction of correct and complete questions (questions with code=4),
S_2	= fraction of complete questions (questions with code=4 or 3),
S_3	= fraction of answered questions (questions with code=4, 3 or 2),
$S_{1\alpha}$	= fraction of correct and complete questions in I-III (questions with code 4),
$S_{1\beta}$	= fraction of correct and complete questions in IV-VI (questions with code 4),
S_4	= fraction of unanswered questions (questions with code=1 or 0),
C	= fraction of correctly answered questions of all questions with an answer (S1 / S3)
CI_1	= citation index summed over MedLine 1987-1992, if first author,
CI_n	= citation index summed over MedLine 1987-1992, if author,
P_1	= number of publications MedLine 1987-1992, if first author,
P_n	= number of publications MedLine 1987-1992, if author.

Table 3. P-values for the different groups.

	T_c	T_e	S_1	S_2	S_3	$S_{1\alpha}$	$S_{1\beta}$	S_4	C
CI-NCI	0.18 -10.17-53.63	0.09 -3.44-8.38	**0.04** -0.57-0.03	**0.01** -0.52-0.02	**0.01** -0.46-0.00	0.90 -0.46-0.20	**0.01** -0.68–0.10	**0.01** 0.0-0.46	0.06 -0.46-0.08
E-NE	0.50 -20.56-37.18	0.94 -7.67-2.23	0.95 -0.24-0.30	0.40 -0.20-0.26	0.40 -0.21-0.21	0.19 -0.16-0.34	0.41 -0.34-0.29	0.40 -0.21-0.21	0.61 -0.20-0.24
SE-NSE	0.15 -47.91-4.13	0.60 -1.83-7.03	0.96 -0.26-0.28	0.96 -0.25-0.19	1.00 -0.22-0.14	0.98 -0.26-0.28	0.98 -0.31-0.31	1.00 -0.14-0.22	0.91 -0.20-0.26
Md-NMd	0.55 -37.13-19.65	0.08 -0.11-8.79	0.09 -0.53–0.09	**0.02** -0.48–0.10	**0.02** -0.41–0.11	0.98 -0.40-0.04	**0.02** -0.64–0.18	**0.02** 0.11-0.41	0.17 -0.40–0.02

P-value ≤ 0.05 indicates that the values are not from the same distribution, and thus are statistically significant different.

CI/NCI	=	involved in patient care/not involved
E/NE	=	experience of data analysis derived from publications MedLine/no experience
SE/NSE	=	statistical experience as stated in the evaluation form/no experience
Md/NMd	=	having a medical degree/no medical degree
#	=	number of users
T_c	=	time of course (min.)
T_e	=	average time per question answered (min.)
S_1	=	fraction of correct and complete questions (questions with code=4),
S_2	=	fraction of complete questions (questions with code=4 or 3),
S_3	=	fraction of answered questions (questions with code=4, 3 or 2),
$S_{1\alpha}$	=	fraction of correct and complete questions in I-III (questions with code 4),
$S_{1\beta}$	=	fraction of correct and complete questions in IV-VI (questions with code 4),
S_4	=	fraction of unanswered questions (questions with code=1 or 0),
C	=	fraction of correctly answered questions of all questions with an answer (S1 / S3)
CI_1	=	citation index summed over MedLine 1987-1992, if first author,
CI_n	=	citation index summed over MedLine 1987-1992, if author,
P_1	=	number of publications MedLine 1987-1992, if first author,
P_n	=	number of publications MedLine 1987-1992, if author.

Table 5. Subjective evaluation data.

	Compared		Time Savings	
	mean	p-value	mean	p-value
Cl	3.00		2.67	
NCl	2.47		2.27	
		0.15		0.36
E	2.67		2.33	
NE	2.44		2.33	
		0.54		1.00
SE	2.00		1.86	
NSE	2.91		2.64	
		0.002		**0.02**
Md	2.86		2.43	
NMd	2.36		2.27	
		0.09		0.77

Compare	=	Subjective judgement of workstation functionality compared with current support (1=less,2=equal,3=more).
Time Savings	=	Subjective judgement about the time savings of the workstation compared with current support (1=less,2=equal,3=more).
Cl/NCl	=	involved in patient care/not involved
E/NE	=	experience of data analysis derived from publications MedLine/no experience
SE/NSE	=	statistical experience as stated in the evaluation form/no experience
Md/NMd	=	having a medical degree/no medical degree

APPENDIX B

The Introductory Course Medical Workstation

Introduction Course

course The purpose of this course is to familliarize you with the Medical Workstation. You should get to know two aspects:
- What functions can it perform
- How the system should be used.

The system is meant to support physicians with their clinical scientific work. Both functionality and use (user-interface) should help to achieve this.

test At the end of this course, a test is given. This test is intended to see whether you are reasonably familiar with the workstation. After this course you will be asked to do some experiments with the workstation. These experiments help to give us an impression as to how well the system's functionality and use relate to physician's wishes, and to get feedback about any bottlenecks. During this course you can consult your supervisor for any question.

Introduction Medical Workstation

the project The project 'Medical Workstation 2000' started in 1987. The project's aim is to research methods and techniques, in order to give a user-friendly access to existing and frequently used computer programs. Especially the physician with less computer experience should profit from such a user-friendly access.

The first prototype was made to support clinical scientific research in cardiology. This was followed by the development of identical environments for other application areas.

support The workstation's user-interface minimizes keyboard input. It also automatically handles many actions:

- interchanging data between programs
- starting functions of programs
- gaining access to other computers using a network

In most cases the user will not even notice these actions.

Where possible actions are performed with a graphical user-interface; a so-called mouse.
Data analysis consists of certain phases. The following phases will be described:

- database selection
- data selection
- data inspection
- descriptive statistics
- presentation

data The workstation contains several research databases. Any data that logically belong to each other is called a database. The following databases can be accessed using the workstation:

- a database from the Paediatric Sophia Hospital Rotterdam
- IMPETUS, a database which contains a part of the data from the Thorax Centre Utility System (TUS)
- a somatostatin database
- an oncological database

data selection Subsets can be made from these data. Such a selection can be made by:

- choosing variables
- choosing only certain values of a variable.

A selection can be made, for example, by choosing the variables 'age' and 'diagnosis' from the Sophia database. One can then only select those children whose value of the variable 'age' is less than 365 days. After selecting the variables and criteria the data are loaded in the computer's working memory (from another computer, through the network) by the command 'submit'. This phase is called 'data selection' in the workstation, and can be executed by choosing 'New data selection'.

data-set The workstation transforms these data into a 'data-set'. The data are now present in the workstation and ready to be used. Names of the database and data-set can always be found in a rectangle at the bottom of the screen.

data inspection After having selected the data, they can be inspected. This inspection is automatically started after the selection phase. While in the inspection phase, data can be checked. Minimal and maximal values of variables can be looked at. One can inspect the distribution of values, or one can just 'go through' the data. Inspection of the data is necessary to see whether the selected data are indeed those that we intended to select in the first place. Data inspection can, at any time during a data analysis, be performed by choosing 'Inspect Current Selection'.

descriptive statistics After the data inspection, several descriptive statistical operations can be performed on the data-set. Again, most statistical procedures can be executed by just clicking the mouse-button once. The data-set will be automatically translated into a form needed by the statistical program.

presentation The result of most statistical procedures can be visualised. These graphics can be looked at and changed with the program Harvard Graphics. This graphical presentation program is available in the workstation under the name 'Harvard Graphics'.

A paper printout can be made during each phase.

Graphical interface and mouse-control

graphical interface The Medical Workstation has a graphical user interface. This means that computer output (what you see on the screen) is not only limited to alphabetical characters, and input is usually done with the mouse. Functions can be called and results can be shown by means of this interface. It "hides" the real computer commands, while the user only sees the names of the functions. One of these functions, for example, is called 'contingency tables'. This function can make cross-tables or frequency tables from the data. When a user chooses this function, he will see only the result. What the computer really does is this: It selects the needed data, it transforms the data into the required form, executes the program that can calculate the tables and eventually shows the table.

mouse The most important tool for operating the workstation is the mouse. It can be moved on the table. The computer then moves an arrow (cursor) on the screen. Cursor movements correspond to mouse movements. The mouse has two buttons. The workstation only makes use of the left button. If we want to choose a name, we have to place the cursor on that name, and then press (or click) the left button. Clicking is necessary to let the computer know that we have not accidentally placed the cursor on a name, but that that name was the goal of our movement.

screen saver Should the screen turn completely black, then the screen-saver has become active. This program makes sure that the screen is not active unnecessarily. Moving the mouse makes the screen active again.

keyboard Sometimes, however, the keyboard is still necessary for input. It is, for example, the most suitable way to type a name. Whenever this is needed, the keyboard will be available for use. These cases will be clearly described in the tutorial.

Course

data analysis We will now go through some stages of data analysis, to give you a good impression of the possibilities of the workstation.

logging in The workstation is a multi-user environment. This means that several users can use the computer at the same time. First of all, a user has to 'log in'; he is asked to give his name and his "secret" password. Access will only be given if they are both correct.

window On the screen you will see a list of names surrounded by a rectangle. A rectangle like this is called a 'window'.

scroll-window If you now move your cursor to the bottom arrow, and press the left mouse-button, you will see that the list of names goes downward. The list goes upward if you place the cursor on the top arrow and press the left mouse-button. Another way to move the list is pressing the left mouse-button while you have placed the cursor on the block between the two arrows. Moving a list up or down is called scrolling; therefore this is a scroll-window.

scroll-bar The rectangle you have just used to scroll through the list is called a 'scroll-bar'. Another scroll-bar can be found at the bottom of the list. This scroll-bar can be used to scroll a text to the right or left. If you see the name 'Tutorial 1', select it using the mouse.

Select the name 'Tutorial 1'.

You will now see a rectangle underneath the list of names. This window should be used to type in the 'secret' password. Move the mouse to this window and press the left button. A 'text cursor' will appear. Any time a cursor like this appears, it means that you will have to type something using the keyboard.

password You can now type the secret password 'tutorial'. Note : because this is a secret name, you will not see what you have just typed! When you have finished typing the password, press <RETURN>.

Type the secret password 'tutorial' and press return.

The workstation checks whether the chosen name matches the access name. If it doesn't you have probably made an error while typing. A window with a message will appear. If it does, select 'OK' with the mouse, and the message will disappear. Then reselect the name 'Tutorial 1' out of the list of names.

log in information After having typed the access name correctly some lines will appear in the window. These give information about:

- where the programs are
- when you were first known to the system
- when your permission to use the system ends
- which type of user you are
- etc.

If you agree with the text you can select 'Accept'.

Select 'Accept'

initialisation The workstation will now be working for some time. All the necessary knowledge for using networks, programs and data are being read and loaded into memory. This will take some time. Meanwhile text-lines will flash on the screen. When the system is ready, a few lines will be written on the screen. Construction and purpose of the system will now be explained in a brief intermezzo.

Intermezzo

lay-out The user-interface is built up as follows. On screen, the top line consists of names: About, Data, Analysis etc.. This line is called the 'menu-bar'. Move the cursor over to this line. Each word is a button. If you click the left mouse-button while on a word, a new list of names will appear. You have just chosen a function. The cursor becomes hand-shaped.

drop-down menu This is called a drop-down menu. So long as a drop-down menu is on-screen the cursor will be hand-shaped. There are two ways to make this drop-down menu disappear. Firstly by placing the cursor outside the menu and clicking the left mouse-button, and secondly by selecting another function (that is: placing the cursor on another word on the top line). In the latter case, the old menu will disappear, while the menu belonging to the newly chosen function will appear. It is also possible to keep the left button depressed, and then move the cursor over the menu-bar. Now menus will appear and disappear. An arrow on the right- hand side of a name in a menu means that there is another, new drop-down menu on that position. Every name in a menu without such an arrow is a function.

Summarizing, there are two ways to select a function:

- by choosing a drop-down menu and then selecting a name. This means clicking the mouse button twice.
- by pressing the mouse button (and keeping it depressed), move to a function in the menu-bar, move to a function in the drop-down menu, and THEN releasing the mouse button. So in this case we have pushed the button only once, while releasing it meant that we were making a choice.

status-bar At the bottom of the screen, a line can be found which contains messages and indicates the system's state: the status bar. Here you can see who is using the system, which database and data-set are currently in use and what the system is doing.

working space The area between menu- and status-bar is the working space. All windows created by programs will be shown in this area.

active windows The window's colour indicates whether or not the window is active. Green means active, yellow not active. A window can be activated by placing the cursor on it. Keyboard input can only be performed in active windows.

Example 1

database selection Choose 'Data' in the menu-bar. You will now see a list containing functions. The function 'New database' has another drop-down menu. In this drop-down menu you can see where the data come from. From the list of functions under 'Data' you should select 'Select Database'.

Select 'Data' in the menu-bar.
Select 'Select Database' in the drop-down menu.

This function enables you to select the databases you want to use. After you have chosen this function, a message appears in the status-bar, after the word 'status'. It gives information about the status of the function. A new window will appear on the screen after some seconds. It contains a list with all the possible databases you can choose from. Select the database 'Somato'. A description of this database can be obtained by selecting the button 'Give Information'. A new window appears with the description and information about who has made the database. This information-window can be exited by selecting 'OK'.

Select 'Give Information'
Select 'OK' in the information-window

The Somato database contains information about somatostatin-scans. During each scan several variables have been measured. Choose 'Select' in the database selection program. Choose 'OK' in the 'current research setting' window. If you now look behind the word 'database' in the status-bar you will see that the name of the database has been changed.

Choose 'Select' in the database selection program
Select 'OK' in the next window

selection from database:

data-set The data-set is now put to 'unknown'. A data-set is a selection from the database. One can select the variables in which one is interested (not all the variables in the database are necessary), or one can select only a number of patients (not all the patients meet certain criteria).

existing data-sets If you had made a data-set during a previous session, then you now could have selected 'select data-set' in the menu under 'Data'.

new data-set Because this is the first time you are working with this database, no data-sets are available. Therefore you will now have to start creating a new data-set. This is done by selecting the function 'New data-set' under 'Data'. This function enables you to make a data-set from the database by just making selections. Usually databases use cryptic names for their variables. While using the medical workstation you will see that all variables have specific medical names.

Select 'New dat-set' under 'Data'

frequency table The first thing we would like to know is what the frequencies are of certain diagnoses in our population. No restrictions are made. Select in the 'Sqlselect' window the function 'Variables' and in the drop-down menu the function 'get Variable'.

Select 'Variables' in the sqlselect program
Select 'get Variables' in the drop-down menu

This function reads the medical names of the variables from a 'dictionary'. Some data can be grouped. Administrative data for example are grouped under the name 'Administrative data'.

relation Such a group of data is called a relation. The somatostatin database uses only one relation, namely: Somatostatin research. If you choose this relation (by selecting its name) you will see a list containing the variables/data that belong to this relation. You can now select the variables that you would like to use. In this case only select 'diagnosis'. Go to 'Actions' and then choose 'Select'. The variable 'Diagnosis' will appear in the first window. Now go to the first 'sqlselect' window, and select the function 'Submit' which you can find under 'Options'. This function makes the workstation create a program which is to be used by the database system. It is then sent to the database.

Select the relation 'Somatostatin research'
Select the variable 'Diagnosis'
Select the function 'Select' under Actions'
Select, in the 'sqlselect' window, the function 'Submit' under
'Options'

other computers This program is then sent, through a network, to the computer where the database is stored. This task will take some seconds. Meanwhile, the workstation will show a message saying that it is busy getting the data. If this message appears then select 'OK' in the 'Message' window.

Select 'OK' in the 'Message' window

data inspection

After some time a small window containing the selected variable will appear. The window's name is 'Inspect current selection'. The data can be presented as a table or as a frequency distribution. Missing values can be detected and simple descriptive information like mean and median are given. This graphical inspection possibility is started automatically after the data are read. However, this function can also be started at any other time by selecting the 'Inspect current selection' function which can be found under 'Analysis' in the menu-bar.

inspection of data presented as a table

Select in the 'Inspect current selection' window the variable 'Diagnosis' and select under 'Options' (in this window) the function 'Table'.

Select the variable 'Diagnosis'
Select the function 'Table' under 'Options'

A window will appear in which the data are presented as a table. You can go through the data using the scroll-bar. After the data inspection, the window can be closed by selecting the variable name above the table. Now, select the function 'Exit' under 'Options' in the 'Inspect current selection' window. A new window named 'DATA-SET SAVER' appears.

Select the variable name above a column of the table
Select the function 'Exit' under 'Options' in the 'Inspect current selection' window

data-set saver

This function translates database data into the workstation's general storage structure. Because of this structure, data can be sent to other programs in a fast and easy way. This local storage structure is called a data-set. You can give the data-set a name, which is done by typing it. The name can have up to 14 characters. Special characters (like '@') are not allowed. A suitable name would for example be 'sms001'. Typing <RETURN> after the name has the same meaning as selecting the 'Save' button with the mouse.

Type the name that you would like the database to have
Either type <RETURN> or select the 'Save' button

default action If a button has an additional outline, like the 'Save' button here, then in general this is the default action. This function can also be executed by pressing the <RETURN> key.
The 'DATA-SET SAVER' window disappears. Three windows will remain, namely the 'sqlselect','relations' and 'Somatostatin research' window. We will now show you a special feature of the workstation. A window always has a title-bar. This is the rectangle with the window's name at the top of each window. In the top-right corner of the window two squares are delineated on the right-hand side of the title-bar.

iconising One of the squares contains a dot. If you select this square, then the window will be replaced, as a small picture, to the bottom-right corner of the screen. A picture like this is called an icon. The window will reappear after clicking on the icon twice, quickly after each other. In general, any window that has a square with a dot in the top right corner, can be 'iconised'.

Select the square with the dot, in the 'sqlselect' window.
After having placed the mouse cursor on the icon, push the mouse button twice quickly

enlarging to screen-size

Next to the square with the dot, is a square with another square in it. When you select this square, the window will enlarge to the size of the screen. Reselecting the square will return the screen to its original size. Now, iconise and de-iconise the window. Enlarge the screen to screen-size, and bring it back to the original size.

Select the square with the square in a window
Reselect this square in the enlarged window

moving a window Windows can also be moved. This is done by selecting the rectangle that contains the name, keeping the mouse button depressed while moving the mouse.

Select the window's title-bar.
Keep the mouse button depressed.
Move the mouse.

enlarging and reducing a window

There are more possibilities than the above mentioned, to change the size of a window. If you slowly move the mouse over the edge of a window, you will see that the arrow will briefly change into some other shape. The shape depends on whether you move the mouse over an edge or a corner. The window's size can be changed by depressing the mouse button just when the arrow has changed its shape. Keep the mouse button depressed, move the mouse and release the button. Try this on every edge and every corner.

Move the mouse slowly over the window's edges and corners
Press the mouse button as soon as the cursor changes
Move the mouse

window-menu There is one square left to explain: The square with the line on the left side of the title-bar. Selecting this square will make a menu appear. Several functions are listed in this menu, the most important one being 'Close'. This function, will close a window. Try it! The window will disappear. Usually, within a function, there is also a button containing a text like 'Quit' or 'Exit'. Using those to close the window is easier and preferred to using the 'Close' function.

Select the square with the line
Select the 'Close' function in the drop-down menu

overlapping windows If a window is partly covered by another, then there are two ways to make the covered window visible:

- iconise the front window
- Select a visible part of the covered window. This will make the window come to the front.

frequency table We will now continue the analysis. We would like to know what the frequencies are of the diagnosis in our data-set. To get an overview we will have to select the function 'Contingency tables' which can be found under 'Analysis'. Using this function, both frequency- and cross-tables can be made.

Select the function 'Contingency tables' under 'Analysis'.

determinant A new window appears. This window allows you to select a determinant. The row-variable is the determinant. This data-set contains only one variable, namely 'Diagnosis'. Select this variable. Then, select the function 'Create frequency table', which can be found under 'Actions'.

Select the variable 'Diagnosis' as the determinant
Select the function 'Frequency table' under 'Actions'

After the calculations are finished, a new window will appear, containing the table that we have asked for. The window is a scroll-window. Therefore we can use the scroll-bar to scroll through the table. Instead of scrolling, it is also possible to enlarge the window, so that we can see a larger part of the window at once. This reduces the need for scrolling.

zero-rows

Use the scroll-bar to move to the top of the list. Because the table function considers all possible values that can be given to the variable 'Diagnosis' in the database, it may be that some diagnoses do not exist in our selection. These are called 'zero-rows'. They can be removed by selecting the function 'Remove Zero Rows', which can be found under 'Defaults'.

Select the function 'Remove Zero Rows' under 'Defaults'.

histogram

The table can also be presented graphically. Select the function 'Histogram' under 'Graphics'. A new scroll-window will appear, in which the table is presented as a histogram. Make such a histogram and then close this histogram window.

Select the function 'Histogram' under 'Graphics'
Select the function 'Quit' under 'Quit' in the histogram window

pictures for Harvard Graphics

It is also possible to create a histogram that can be used by the program 'Harvard Graphics'. This MS-DOS program has many features to create and adapt histograms, pie-charts etc.. After selecting the function 'Harvard Graphics', a window will appear where you will have to name the picture that is going to be created. By giving different names, several pictures belonging to one data-set can be saved. Make sure that the cursor is placed on the window that is asking for the name (you are on the window if the colour of the window's border differs from that of the other windows). Name the picture. Use up to 14 characters, special characters are not allowed. Select 'OK' or press the <RETURN> key. Move the 'Contingency table' window, so that it is possible to iconize the 'cross' window.

Select the function 'Save for Harvard Graphics' under
'Graphics'
Name the picture
Select 'OK' or press <RETURN>
Move the 'Contingency table' window and iconize the 'cross' window

Harvard Graphics

Now select the function 'Harvard Graphics' which can be found under 'Presentation' in the menu-bar.
A scroll-list appears containing the names of all the existing scripts of Harvard Graphics pictures belonging to this data-set. Select the name that you have just given to your picture and then select 'Use Script'. You will now have to wait some seconds because an MS-DOS PC is going to be emulated on the UNIX workstation. Then a window will appear in which this emulated PC is going to be started. Just let the station work.

Select the function 'Harvard Graphics' under 'Presentation' in the
menu-bar
Select your script out of the script list
Select 'Use script'

All the necessary information will automatically be given to Harvard Graphics by the workstation. The histogram will appear without having to give commands or type names. Harvard Graphics can now be used as if it is running on an MS-DOS PC. By pressing the <ESC> key, for example, data can be changed. We could for example change the texts below the histogram, because they overlap. But, because this course is not intended to teach you how to use Harvard Graphics, we'll leave the texts this way and just continue. Press the <ESC> key twice. Use the arrow keys to go up to the function 'Create New Chart'. Press <RETURN>. Use the arrow keys to go to the 'Pie' option and press <RETURN>. Harvard Graphics will then ask whether we want to use the same data. Because we do, press <RETURN>. Then press the key marked <F2>, and we'll see a pie-chart instead of a histogram. This way we are able to make all kinds of graphs of the results of our analysis. We now end our first example analysis and select the 'Close' function which can be found in the drop-down menu after you have selected the square with the line.

Press the <ESC> key twice
Use the arrow keys to go to the function 'Create new chart'
Press the <RETURN> key
Use the arrow keys to go to the function 'Pie'
Press the <RETURN> key
Press the <RETURN> key
Press the <F2> key
Select the square with the line
Select the 'Close' function

A second example We've just finished the first example, but we'll give you a second example. You may, and hopefully you will, also try out other features of the workstation during this example.

Example 2

data selection

Select the 'Contingency table' icon and close the 'Contingency table' window. Select the function 'New data-set' under 'Data'. Select the function 'Get Variable' under 'Variables', and select, in the 'Somatostatin research' window, the variables 'Diagnosis', 'Histology result', 'Sex', 'Alpha-subunit measurement' and 'Scan result'. Select 'Select' under 'Actions'. Then close all the windows, except for the 'sqlselect' window, by selecting 'Exit' under 'Actions'. Select 'Unmark' under 'Variables'.

Select the 'Contingency table' icon and close the window
Select the function 'New data-set' under 'Data'
Select the function 'Get Variable' under 'Variables'
Select the variables 'Diagnosis', 'Histology result', 'Sex',
  'Alpha-subunit measurement' and 'Scan result'
Select 'Select' under 'Actions'
Close all the windows, except for the 'sqlselect' window
Select 'Unmark' under 'Variables'

conditions

You are now going to set some selection-criteria for this data-set. First of all, we want only those patients whose diagnosis has been confirmed by histology. Select the variable 'Histology result' out of the list and select the function 'Enter search condition' under 'Patients'. A new window appears. If you select the condition 'is equal to', then you will see the other possible condition, namely: 'is not equal to'. The workstation 'knows' that this variable can only be equal or not equal to something (and not, for example, be smaller than something). If you select 'is equal to', and then move the mouse, while keeping the mouse button depressed, to the other possible value, then the workstation will assume that you have selected the condition on which you have released the mouse button. Select 'confirmed'. Move the mouse to 'not confirmed' and release the mouse button there. The button will now say 'not confirmed'. Reselect this button and put the value back to 'confirmed'. Select 'OK', and you will see that the list now holds the first condition. Now, reselect the variable 'Histology result' in the top list, although this variable is already selected. Reselecting it will in fact deselect it. Select the variable 'Sex' and select 'Enter search condition'. Indicate that you are only interested in male patients.

Select the variable 'Histology result' out of the list under
'I want to see..'
Select the function 'Enter search condition' under 'Patients'
Select the condition 'is equal to' 'confirmed'
Select 'OK'
Reselect the variable 'Histology result'
Select the variable 'Sex'
Select 'Enter search condition'
Select 'Is equal to' 'Male'
Reselect the variable 'Sex'

combining conditions

A second condition line has appeared in the lower scroll-list. On the left side of this scroll-list you will see a button saying 'OR'. This means that patients either meet condition 1 or 2, or conditions 1 and 2. But what we really want is only those patients who meet both conditions. Therefore we would like to change the 'OR' into 'AND'. This is done by selecting the 'OR' button. Now deselect the variable 'Scan result' and select 'Diagnosis'. Furthermore, we are only interested in those patients whose diagnosis is either 'Apudoma', 'Carcinoid', 'Tumour of the lung' and/or 'Hodgkin's disease'.

Select the 'OR' button
Select the variable 'Diagnosis'
Select those patients whose diagnosis is either 'Apudoma', 'Carcinoid', 'Tumour of the lung' and/or 'Hodgkin's disease'

grouping conditions

After you have entered these conditions, a problem has occurred. We are not interested in the patients who have all the diagnoses (if they exist), but in those patients who have one of these diagnoses. To overcome this problem, you select in the condition list all conditions that relate to a diagnosis. Then set the button to 'OR' and select the function 'Group' under 'Patients'. By doing so, all diagnosis conditions will be grouped in one expression. The window that now appears offers the possibility to name the grouped condition. Just select 'OK'. You will now see that there is only one line left with conditions relating to diagnoses. Set the button to 'AND', and send this command to the computer where the database resides. This is done by selecting the function 'Submit' under 'Options'.

Select the conditions that relate to diagnoses
Select the 'AND' button (becomes 'OR')
Select the function 'Group' under 'Patients'
Select 'OK'
Set the button to 'AND'
Submit the selection command

data inspection

Select 'OK' in the 'Message' window after this window has appeared. Inspect the data which will be presented as a table in the 'Inspect current selection' window. To do so select all variables and select 'table' under 'Options'. Now deselect all selected variables and select, once again, the variable 'Alpha-subunit measurement'. Select the function 'Histogram' under 'Options'. You will now get a good impression of the distribution of the data in this data-set. Select 'Quit'. If you wish you could now make histograms using other variables.

Select 'OK' in the 'Message' window
Select all variables in the list
Select 'table' under 'Options' in the 'Inspect current selection' window

Select the name of a variable above a column in the table
Deselect all selected variables
Select the variable 'Alpha-subunit measurement'
Select the function 'Histogram' under 'Options'
Select the function 'Quit' in the latest window
Repeat this histogram function using other variables

data-set saver When you have finished the inspection, select the function 'Exit' under 'Options' in the 'Inspect current selection' window. Name the data-set in the 'Data-set saver' window. Then iconize the 'sqlselect' window and select the function 'Contingency tables' under 'Analysis'.

Select the function 'Exit' under 'Options' in the
'Inspect current selection' window
Name the data-set
Iconize the 'sqlselect' window
Select the function 'Contingency tables' under 'Analysis'.

cross-table Because several variables have now been selected, the 'Contingency table' window will let you select as well a determinant (=rows) as well as an outcome (=columns).

outcome The outcome is the variable in the column.

determinant Select the variable 'Alpha-subunit measurement' as determinant and 'scan result' as outcome. Select the function 'Create cross table..' under 'Actions'. In the window that now will appear, we'll have to group the values of the variable 'Alpha-subunit measurement'.

Select the variable 'Alpha-subunit measurement' as the determinant
Select the variable 'scan result' as the outcome
Select the function 'Create cross table..' under 'Actions'

cut-off points

The variable 'Alpha-subunit measurement' is a continuous variable. You will have to indicate the cut-off points. There are two ways of doing this. Either you move the slider directly underneath the graph to the left or right, or you can use the field under this slider. If you use the slider then, after you have chosen a certain value, you will have to select the function 'Accept cut-off' under 'Accept' to confirm the cut-off point. If you use the field then you can use the keyboard to type the desired value. You take the second option and type the value 1.1. Now, select 'Accept'. A vertical black line will now appear in the distribution graph. Although we could select more than one cut-off point, one suffices for the example. A value up to 1.1 is the normal value of an 'Alpha-subunit measurement' in healthy men. If you now select 'Quit' then a table will be presented in which the 'Alpha-subunit measurement', divided into normal (<=1.1) and abnormal (>1.1), will be plotted against the 'scan result'.

Type 1.1 to indicate the cut-off value
Select 'Accept'
Select 'Quit'

confidence interval

What we would like to know now is whether the distribution of scan results in the 'Alpha-subunit <=1.1' group differs significantly from the distribution of results in the 'Alpha-subunit >1.1' group. Therefore we have to calculate the confidence interval. This is done by selecting the function 'Contingency statistics' under 'Analysis'. Confidence interval and p-values will now be presented in a window. If you like you could also try other methods of calculation or other confidence limits. After this, close the window.

Select the function 'Contingency statistics' under 'Analysis'
Also try other methods of calculation and calculations using other
confidence limits
Close the window by selecting 'Exit'

histogram There are some more default settings that can be changed under 'Defaults'. Under 'Transform' more actions can be found, which can be carried out on the table. These actions include deleting and grouping categories. Histograms can also be made. Every table can be printed out on paper by selecting the function 'Print'. Ask the supervisor for the location of the printer. Close all the windows, including any iconized window.

Select 'Print'
Close all windows

test We hope that these two examples have made you sufficiently familiar with the workstation's potentials. We'll now continue with a small test in which you will be asked to answer two questions. This will be followed by the evaluation experiment.

Concluding test

Question 1

data selection Select the 'Somato' database. We would like to make a new data-set out of this database. Select the variables 'Diagnosis', 'Age' and 'Histology result' out of the 'Somatostatin research' group. We're not interested in all possible values, so we're going to add some selection conditions (by selecting the 'Unmark' option under 'Variables'). The 'Diagnosis' has to be equal to 'Carcinoid' AND the 'Histology result' has to be 'Confirmed'.

data inspection Retrieve these data (Submit) out of the database. Inspect the data, using the data inspection program, for example by making a histogram of the variable 'Age'. Write down the mean and median of the variable 'Age'

Mean age:
Median age:

cross table Close the 'Inspect current selection' window and save the data-set. Use this data-set to make a cross-table in which the 'Diagnosis' is the determinant, and age is the outcome. The patients have to be divided into two groups: patients whose age is 50 years or less, and patients older than 50 years. Remove the zero-rows in the table. Write down the number of patients younger than or equal to 50 years AND diagnosed as having a 'Carcinoid'. Also write this number down for patients aged over 50 years.

Carcinoid and 50 years or younger:
Carcinoid and over 50 years:

histogram Make a histogram using these data and make a paper print. Close all opened windows, if any, leaving the screen in the starting state.

Question 2

data selection The Somatostatin database is used again to make a new data-set. Select the variables 'Alpha-subunit measurement', 'scan result', 'Age', 'Histology result', 'Diagnosis' and 'Sex'. You only want patients whose diagnosis is one of the following: Apudoma, Carcinoid, Tumour of the lung or Acromegaly. (Group using 'OR'). The patients should also meet the following conditions: Confirmed histology result, male, positive scan result. (Use 'AND'). Retrieve the data.

data inspection Write down the mean and median age.

Mean age:
Median age:

cross table	Make a cross-table in which the age is plotted against the Alpha-subunit measurement. Let the cut-off point of the Alpha-subunit measurement be the normal value 1.1. Use four age groups of which the cut-off point are 25, 50 and 75 years. Create the table and make a paper printout.
histogram	Calculate the confidence interval. Create the corresponding histogram and make a paper printout.
ready	Close all existing windows. This is the end of the test.

APPENDIX C

MW2000 - A workstation for the support of clinical research and patient care in cardiology. Part I: Design considerations and Current Status

T. Timmers, E.M. van Mulligen, F. van den Heuvel, J.H. van Bemmel

Department of Medical Informatics, Erasmus University Rotterdam, The Netherlands

Abstract

The full potential of computer applications in clinical research and practice is not yet in the hands of the clinician. There are two main reasons for this unfortunate situation. First, most existing systems have been developed separately and cannot communicate with each other. Second, individual programs have widely differing user interfaces which are usually not medically oriented. The MW2000 Workstation aims at solving these problems by offering a user-friendly integration of databases, statistical packages and other programs in a network for the integrated support of clinical research and patient care. Its flexible architecture allows the smooth integration of a wide range of software packages and other resources.

Introduction

Monday night, 8.00 hrs p.m., Department of Cardiology. Assistant X has found some time to continue his research study on survival of hypertrophic myocardiopathy. While at home during the weekend he has updated his database of some 100 patients with the latest findings on mortality from the municipality and findings from the paper medical record. The update must be imported into the Workstation for analysis. He has brought the dBaseIII files with him, logs onto the Workstation and gets access to his own research directories. After typing in the name of the file, the database is copied into the Workstation research database, and the analysis can start. First, X makes a selection from the database of the variables and patients related to follow-up (Figure 1). The variables are presented on the screen by their usual medical name; and conditions are entered by selecting from code lists that are stored in the system, so X does not need to remember them. The selections are made by means of a "mouse", a device that allows one to point at spots on the screen. Subsequently, a contingency table is created of "Syncope during Follow-up" against "Status" (i.e., alive or deceased) (Figure 2). By simply "pointing and clicking" with the mouse at one of the appropriate "menus", a program is started that computes and shows the corresponding P-values and rate ratios (Figure 2). Because X is to present his results at the next staff meeting, he uses another Workstation facility to prepare slides of the contingency tables in Harvard Graphics (not shown here). Again, this is done automatically, without the need for retyping the data.
In the earlier intermediate analyses, the mortality rates were surprisingly low. X is interested to know what a formal survival analysis reveals. X selects the "Life table and Survival analysis" option from the "Analysis" menu, and the Workstation automatically starts up module 1L of the BMDP Statistical Package. Subsequently, the Workstation graphically presents the result on its screen (not shown here). X decides to consult the Department's statistician Y for further analysis, but right now he makes a printout of the graphs. Maybe Y needs to make a more extensive analysis of the data, using the full functionality of BMDP. At least these initial results can be communicated to Y. X drops a message in Y's electronic mailbox, which Y will read next morning when he logs onto the Workstation through the terminal on his desk.
After "playing" with some other "views" on his data, X decides to stop and go home. It is Monday night, 9:00 hrs p.m..

A future fantasy? Not at all. At the Department of Medical Informatics, this clinical research is currently done on the Medical Workstation 2000, and its principal researcher is being trained to use the Workstation [1]. The MW2000 has more to offer, including, e.g., facilities for the display of biomedical signals and images, such as ECGs and angiograms (e.g., Figure 2).

More facilities could be mentioned, and undoubtedly the reader can suggest extensions supporting his particular needs. However, rather than going into more detail, let us step back and look at how computers have been used for the support of clinical research and patient care until now, and how this might radically change in the near future by the advent of Workstations such as the MW2000.

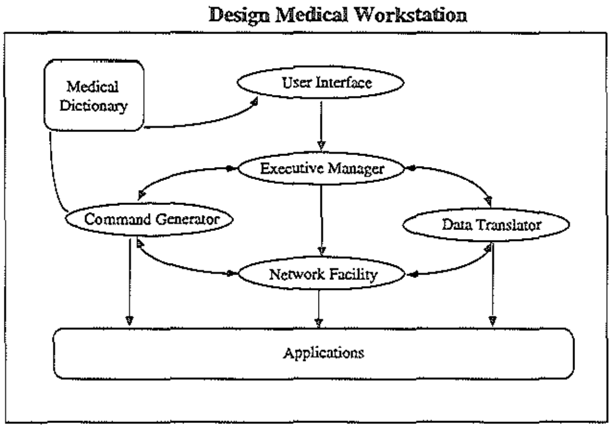


Figure 3. Software architecture of the MW2000, showing its basic components.

1. User-Friendly Interfaces and Integration of Resources

Over the past 25 years, Medical Informatics has matured to the point where several well-established and validated computer programs, systems and methods exist. For example, in Cardiology methodologies and systems exist for the storage and retrieval of patient data (departmental and hospital information systems); for signal processing (ECG analysis and interpretation) and image analysis (assessment of obstructions in arteries). Recently, systems have been developed for the purpose of intelligent consultation.

The ideal is to be able to combine the specific capabilities of these systems, and maximally profit from the many years of effort spent in their development. In practice, there is a need to easily retrieve data from a Hospital Information System (HIS), and to combine those data with those from a Departmental Information System and other sources.

However, in clinical research and patient care one is confronted with the fact that these systems often run on different types of computers, which are also not interconnected. This entails laborious transfer of data from one system to another; moreover, programs also operate in widely differing ways. Most user interfaces are not immediately understandable, and the novice user needs much time to acquaint himself with a specific program.

Several authors identified the solution to these problems as paramount to the next generation of computer support in medicine. Greenes [2] points at modularity, sharing, and integration as the key issues. He describes the modern clinical user of computer resources as a typical "problem-solver", who needs to be able to quickly move from one resource to another, rather than limiting himself to just one task.
Currently, several research groups are developing approaches to alleviate these problems. For example, some deal with institution-wide integration using networks such as in the IAIMS project [3], others aim at building various kinds of connections between resources [4]. Many of these projects require that existing systems and programs are rebuilt to make integration possible.

In contrast, the starting point of the MW2000 project is to use existing software avoiding the necessity of costly adaptation. It aims at developing and implementing a new Workstation architecture which offers a user-friendly, medically-oriented interface, and consistent integration of existing software and other resources. We will discuss what this means, using the example of the research described in the introduction.

User-friendly interface
User-friendliness in the MW2000 works at various levels. For starting-up a particular computer program, the user only needs to "point and click" at the appropriate entry in one of the menus. The user need not remember the name of the program. This level of user-friendliness is well-known, e.g. in the world of MS-DOS this is achieved by the MS-Windows environment.
A more powerful level of user-friendliness is shown by the survival analysis user-interface. Because of its reliability, and because it was already in use at the Thoraxcentre, BMDP was chosen as the statistical package. Any other statistical package can easily be incorporated. In the Workstation, the user need not know anything about the relatively complex command language that BMDP employs. Most of its options can be entered

through clicking at specific fields, just as one ticks entries on a form. The appropriate command language is generated automatically and sent to the appropriate module of BMDP, which may run on any other computer in the network, without the user being aware of it. Subsequently, the tabular output is interpreted and redrawn.

Medical orientation through a medical terms dictionary

In the above example, the user selects variables by choosing from a list with the full medical names of those variables. The user need not know that "Effect of Betablocker on LVAO gradient" is represented in the database by the attribute name "LVAOBB". Nor does the user need to remember that in this case, the code value "1" signifies "increase". The Medical Terms Dictionary takes care of the translation of the name. In the near future, this dictionary will also contain information such as valid and plausible ranges of variables.

Integration of existing programs and resources

This aspect is also expressed at various levels. Integration is achieved at the level of the institution by offering transparent access to other systems through a network. For example, at the Department of Medical Informatics, BMDP and Ingres run on machines other than the MW2000 itself, but the user remains unaware of that.

Integration at yet another level is achieved by "encapsulating" different programs and packages and offering automatic data translation between different data formats. This is shown in the above example where data are extracted from the research database Ingres, and automatically translated into the appropriate data format for BMDP. In the MW2000 this is done through the use of a common Intermediate Storage Format. Adding another program to the system only requires addition of another data translator.

The power of this approach is also shown by the integration of Harvard Graphics into the MW2000 (Harvard Graphics is a program which is widely used at the Thoraxcentre for the preparation of graphs, and runs on MS-DOS computers). The Workstation has UNIX as its operating system. However, it is possible to emulate MS-DOS programs in the Workstation UNIX environment. Transporting data from, e.g., contingency tables to the appropriate Harvard Graphics format is done automatically and transparently.

2. Current Status of the MW2000

To summarize, the Workstation currently offers the following facilities:

- o The integrated use of data from various databases such as the Hospital Information System, dBaseIII, Lotus123, etc.;
- o Use of medical terminology rather than idiosyncratic names of variables;
- o Quick visual inspection of the data, contingency tables, descriptive statistics and graphical representation;
- o Support of statistical analysis;
- o Transparent access from the Workstation to programs and data located on other computers;
- o Retrieval and display of signals and images.

The software packages presently available on the Workstation are:
- o Ingres as the central research Database Management System;
- o BMDP as the main statistical package;
- o Harvard Graphics as a graphical presentation package;
- o WordPerfect for text processing;

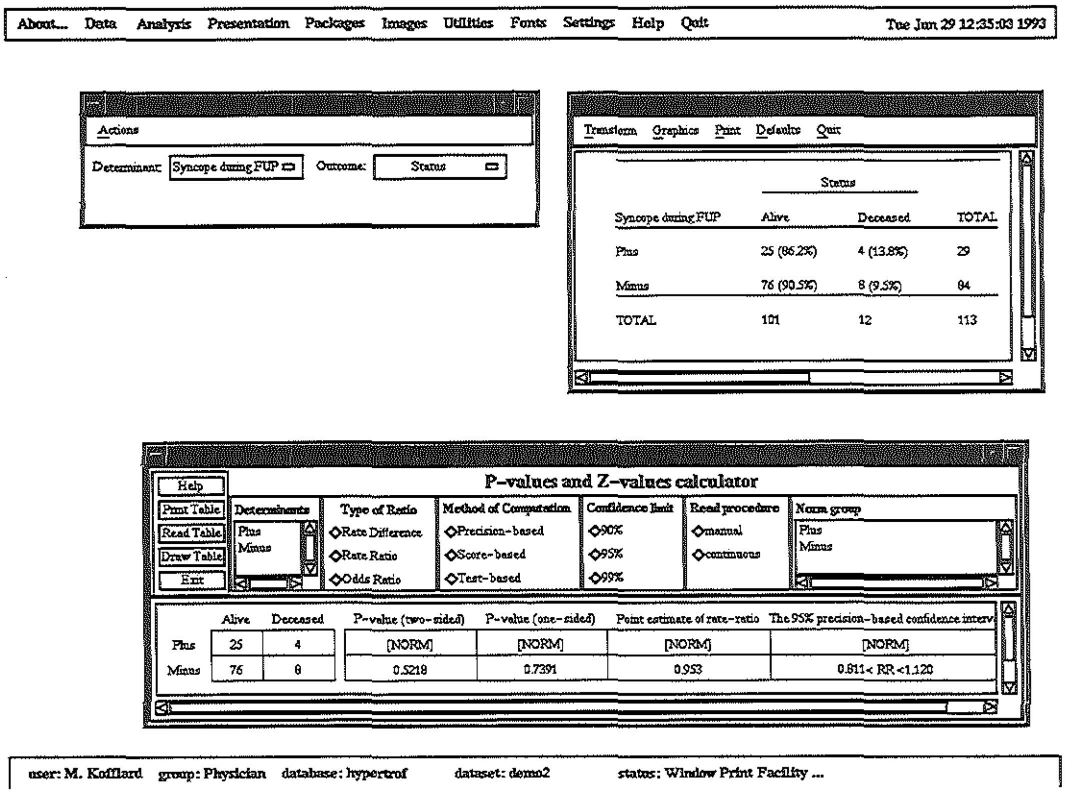
The following packages that have been developed at the Department of Medical Informatics are also available:
- o MEANS, an ECG analysis and interpretation system;
- o ISPAHAN, a package for interactive statistical pattern recognition;
- o KNEW, a multi-purpose knowledge editor developed on the MW2000 for the support of knowledge-based applications, such as medical vocabularies, command-language generation, etc.

In the future, many other software packages will be integrated on the MW2000. The Workstation runs on a HP 9000/380 UNIX graphical workstation with a 17-inch color monitor. Networking is done through Ethernet with the TCP/IP protocol. The user interface has been written using the OSF/MOTIF Toolkit. Recently, a down-scaling to a '386-based system was accomplished, which runs under the SCO-UNIX operating system.
A brief overview of the MW2000 architecture is shown in Figure 3. Details have been

described elsewhere [5]. All user interaction is handled by the User Interface. The options and commands chosen are translated into messages which are sent to the Executive Manager, which controls applications and other facilities. Data format translation is done by the Data Translator, while the Network Facility handles all network communications. The Medical Dictionary allows the user to address the database through familiar medical terms. Finally, the Command Generator creates the proper command sequences, as in the example of the BMDP survival analysis described before.

3. MW2000 Projects

It should be noted that the MW2000 is not only supportive for existing clinical reseach and patient care. It also stimulates a wide range of research and development in medical informatics itself. For completenes, we briefly mention on four examples: user evaluation, further development of the architecture, modelling medical knowledge and terminology, and linking with the HIS [6]. These developments will be discussed in more detail in subsequent issues of the Thoraxcentre Journal.
User evaluation. Currently, a formal user evaluation is done, in which a protocol is defined to objectively asses usability and usefulness of the Workstation. After a tutorial introduction, the user is requested to solve a realistic problem of biomedical data analysis. The results of this evaluation will contribute to the further design of the Workstation.
Architecture. Although already in use, the present version of the MW2000 is basically a prototype. Extensions are needed in the areas of data validation, exploration and statistical analysis. Most importantly, the architecture of the Workstation will be gradually


About... Data Analysis Presentation Packages Images Utilities Fonts Settings Help Quit
Actions
Transform Graphics Print Defaults Quit
Help
Print Table
Read Table
Draw Table
Exit

Tue Jun 29 12:35:03 1993

Determinant: Syncope during FUP Outcome: Status

	Status		
Syncope during FUP	Alive	Deceased	TOTAL
Plus	25 (86.2%)	4 (13.8%)	29
Minus	76 (90.5%)	8 (9.5%)	84
TOTAL	101	12	113

P-values and Z-values calculator

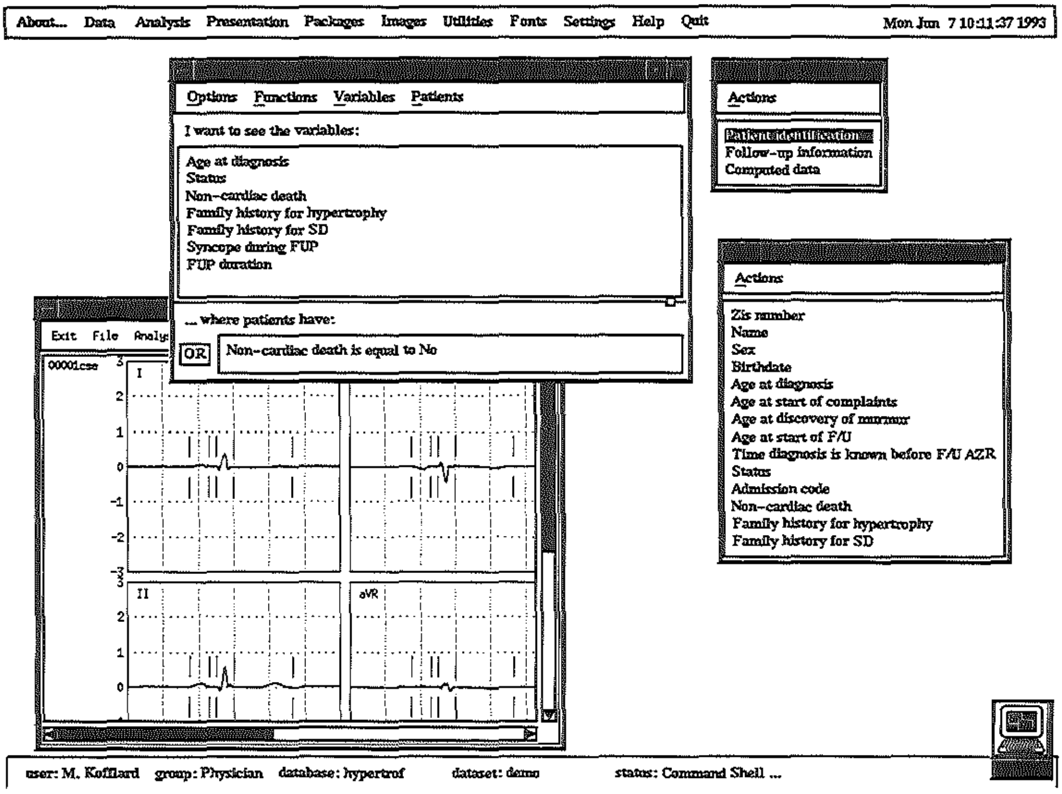
Determinants	Type of Ratio	Method of Computation	Confidence limit	Read procedure	Norm group
Plus	◇Rate Difference	◇Precision-based	◇90%	◇manual	Plus
Minus	◇Rate Ratio	◇Score-based	◇95%	◇continuous	Minus
	◇Odds Ratio	◇Test-based	◇99%		

	Alive	Deceased	P-value (two-sided)	P-value (one-sided)	Point estimate of rate-ratio	The 95% precision-based confidence interv
Plus	25	4	[NORM]	[NORM]	[NORM]	[NORM]
Minus	76	8	0.5218	0.7391	0.953	0.811< RR <1.120

user: M. Kofflard group: Physician database: hypertrof dataset: demo2 status: Window Print Facility ...

Figure 2. Creation of contingency tables. In the upper left window, two variables are chosen, and in the upper right window, the corresponding contingency table is displayed. Below, the P-values and Rate ratios are displayed pertaining to this table.

redesigned to be more consistently based on the client-server model [7]. It seems that this kind of approach is a prerequisite for the much needed institution-wide integration of resources. Plans are being made to introduce the MW2000 on a network into the Thoraxcentre.


About... Data Analysis Presentation Packages Images Utilities Fonts Settings Help Quit
Mon Jun 7 10:11:37 1993
Options Functions Variables Patients
Actions
Actions
Exit File Analy
OR
I
II
aVR

I want to see the variables:

Age at diagnosis
Status
Non-cardiac death
Family history for hypertrophy
Family history for SD
Syncope during FUP
FUP duration

... where patients have:

Non-cardiac death is equal to No

Patient identification
Follow-up information
Computed data

Zis number
Name
Sex
Birthdate
Age at diagnosis
Age at start of complaints
Age at discovery of murmur
Age at start of F/U
Time diagnosis is known before F/U AZR
Status
Admission code
Non-cardiac death
Family history for hypertrophy
Family history for SD

user: M. Koffiard group: Physician database: hypertrof dataset: demo status: Command Shell ...

Figure 1. A Workstation screen showing data selection by means of graphical selection from a list of variables. The top menu bar presents the main menu, while the bottom bar indicates the current status of the Workstation. For data selection, three windows are shown: on the upper right hand side, a group of variables has been slected, and the list of variables appears on the right hand side below. Mouse-clicking on the names of the variables makes them available for selection, as shown in the upper left window.
In the lower left window, an ECG multilead recording is displayed, together with fiducial marks as determined by the ECG interpretation system MEANS.

Modelling medical knowledge. Clinical applications will be more knowledge-oriented. For example, a research project has been started in which the knowledge editor KNEW is used for the development of a controlled vocabulary and support of clinical research in the area of Congenital Heart Disease [8]. This may lead to a better defined standard for diagnoses, examinations and interventions in this complex area in Cardiology, and contribute to a better quantitative analysis of long-term follow-up. Participants in this project are the Department of Medical Informatics, the Division of Pediatric Cardiology of the Sophia Childrens Hospital, and the Thoraxcentre in Rotterdam.

Linking with the HIS. Recently, a project was started in which a flexible link is developed between the MW2000 and the HIS of the Academic Hospital Rotterdam [9]. A pilot project was defined in which data can be retrieved from the HIS and imported into the MW2000 for analysis.

4. Conclusions

Clinical research and practice increasingly take place in an environment where many, widely differing computer systems and software packages are available for the support of the clinician. This implies that there is an increasing need for clinicians to have easy, flexible and integrated access to these software programs and clinical data, images and signals.

Using a network and a powerful graphical workstation, a Workstation architecture has been developed which offers user-friendly, integrated support for clinical research and patient care. One of the prime advantages of our approach is that the MW2000 allows this integration to be achieved with little effort. There is no need for costly adaptation of existing software which is already validated and in use in a particular environment. Our approach is also promising for the much needed institution-wide integration of resources through a network.

In addition, the Workstation has facilities for modelling medical knowledge and terminology. This may lead to, for example, better standardized nomenclatures. These and other developments will be discussed in subsequent issues of the Thoraxcentre Journal.

References

[1] Kofflard M, Ten Cate FJ, Bucx J, Tijssen J. Is hypertrophic cardiomyopathy a more benign disease than previously described? Manuscript in preparation, Department of Cardiology, Thoraxcentre, Academic Hospital and Erasmus University, Rotterdam, The Netherlands.

[2] Greenes RA. Promoting Productivity by Propagating the Practice of "Plug-compatible" Programming. In: *Proceedings of the 14th Symposium on Computer Applications in Medical Care.* Washington DC: IEEE Computer Society Press, 1990:22-26.

[3] Barsalou T. An object-based architecture for biomedical expert database systems. In: *Proceedings of the 12th Symposium on Computer Applications in Medical Care*. Washington DC: IEEE Computer Society Press, 1988.

[4] Lunin LF, Ball MJ (eds.). Perspectives on Integrated Academic Information Management Systems (IAIMS). Journal of the American Society for Information Science, 1988;39(3):102-112.

[5] Van Mulligen EM, Timmers T, Leao BDF. Implementation of a Medical Workstation for Research Support in Cardiology. In: *Proceedings of the 14th Symposium on Computer Applications in Medical Care*. Washington DC: IEEE Computer Society Press, 1990:769-773.

[6] Timmers T, van Mulligen EM, Langhout A, van Bemmel JH. MW2000: Assessment and future developments. Internal report, Department of Medical Informatics, Erasmus University Rotterdam, May 1991.

[7] Van Mulligen EM, Timmers T, van den Heuvel F. A Framework for Uniform Access to Data, Software and Knowledge. In: *Proceedings of the 15th Symposium on Computer Applications in Medical Care*. Washington DC: IEEE Computer Society Press, 1991.

[8] Van den Heuvel F, van Mulligen EM, Timmers T, Hess J. Knowledge-based modelling for the Classification and Follow-up of Patients with Congenital Heart Disease. In: *Proceedings of Computers in Cardiology*, Venezia 1991 (in press).

[9] Timmers T, van Mulligen EM, van Ooijen B. Linking HIS and MW2000; a proposal for a pilot project. Internal report, Department of Medical Informatics, Erasmus University and CDAI, Dijkzigt Academic Hospital, Rotterdam, April 1991.

CURRICULUM VITAE

Erik Matthias van Mulligen was born in Wildervank, Groningen on the 12th of august, 1965. The first five years of his undergraduate education were received at the Christelijk Lyceum Veenendaal in Veenendaal and his final year at the Christelijk Lyceum Buitenveldert (gymnasium ß) in Amsterdam.

In 1987, he received a degree in computer science (with specialty medical informatics) at the Free University in Amsterdam. His final-term project at the department of medical informatics of the Free University was entitled "Reconstruction of the VCG from the ECG". He followed the department to the Erasmus University Rotterdam and joined them in october 1987.

From 1987 until now, he investigated the possibilities of integrated medical workstations as reported in this thesis.

Erik van Mulligen continues his research as a scientific staff member of the department of Medical Informatics at the Erasmus University Rotterdam, and will continue with the HERMES project as presented in this thesis.

DANKWOORD

Ik ben de Heer uw God, die u leert, opdat het u wel ga; die u de weg doet betreden, die gij moet gaan. Jesaja 48:17

Een promotie kan vergeleken worden met het het bouwen van een menselijke pyramide in een circus-act: het succes dat de bovenste man heeft wanneer hij er in slaagt om op de schouders van anderen in evenwicht te blijven is afhankelijk van de kracht en de stabiliteit van de andere mannen in de pyramide. De correcties die zij maken zijn dikwijls niet zichtbaar voor het publiek, maar zijn van essentieel belang voor de balans en opbouw. Ik was erg blij dat ik op de schouders van prof. Jan van Bemmel en dr. Teun Timmers kon staan. Ik waardeer bijzonder het enthousiasme en het vertrouwen van Jan van Bemmel. In de vele gesprekken die wij hebben gehad, heb ik veel geleerd over het beoefenen van wetenschap. Zijn belangstelling, met name ook voor mij als mens, heeft ertoe bijgedragen dat ik met plezier aan mijn onderzoek heb gewerkt. Teun Timmers is altijd bereid geweest om mijn teksten van commentaar te voorzien, en van hem heb ik geleerd kritisch ten aanzien van je onderzoek te zijn. Het contact dat wij hadden, heb ik als zeer prettig ervaren en heeft mede bijgedragen tot dit resultaat.

Daarnaast zijn er mensen nodig die helpen de pyramide bouwen. Zij werken voor een groot deel achter de coulissen, maar hun bijdrage tot het succes is even groot als die van de "sterke schouders".
Drs. August Langhout, ir. Ronald Cornet, Martin Kalshoven en drs. Jaap Brand dank ik voor hun bijzondere bijdrage aan de programmatuur en voor hun zeer grote bereidwilligheid. De inbreng van drs. Freek van den Heuvel en dr. Beatriz de Faria Leao heeft mij bepaald bij de specifieke medische vragen: m.n. hun gedachten over wat artsen nodig heeft de richting van het project bepaald. De stagiares Joke Oudelaar, Francis Lieuw-On, Marcel Koning, Paul Dingemans en Armin Rozema hebben allen waardevolle bijdragen geleverd aan de ontwikkeling van het medisch werkstation.

Voorts zijn er natuurlijk de overige collegas en mensen buiten de vakgroep die allen altijd meegedacht hebben en commentaar hebben geleverd over de ontwikkeling van mijn visie op het werkstation. Tenslotte wil ik mijn ouders, schoonouders en overige familieleden bedanken voor de warme belangstelling die zij altijd hebben gehad. Voor Lydia en de kinderen heb ik diep respect: zij gaven mij zonder vragen, soms zelfs zonder zuchten, de tijd om deze "klus" te klaren. Samen met hen heb ik ervaren dat onze God de enige is die ons de weg kan wijzen en de mogelijkheden geeft om het evenwicht te bewaren.