The Practice of Informatics

How the Past Teaches the Future:

ACMI Distinguished Lecture

W. Ed Hammond, PhD

Abstract More than 30 years of experience in developing a computer-based patient record system, The Medical Record (TMR), in multiple settings, in multiple specialty groups, and at multiple sites has taught us many lessons. Lessons related to computer-based patient records include the importance of a data model in which input, storage, and planned use are independent; separation of patient-specific data from metadata; a modular design to localize the program code that deals with a set of data; redundant storage to optimize tasks and response time; and integration of decision support into work process. Lessons related to medical informatics include the importance of a clinical–technical partnership, control of tools at the leading edge, and rapid prototyping in the real world. Finally, changes in technology move the challenges but do not eliminate them.

J Am Med Inform Assoc. 2001;8:222–234.

Much of my professional career has dealt with the creation, implementation, and support of a computer-based patient record system that became known as The Medical Record, or TMR. In this lecture, I would like to discuss with you some of those experiences—what we did, what the driving factors were, what worked and what didn’t, what lessons we learned, and what implications for medical informatics might be drawn.

Affiliation of the author: Duke University, Durham, North Carolina.

This lecture was originally presented at the AMIA Annual Symposium, November 4–8, 2000, in Los Angeles, California.

Correspondence and reprints: W. Ed Hammond, PhD, Box 2914, Duke University, Durham, NC 27710; e-mail: <hammo001@mc.duke.edu>.

Received for publication: 1/19/01; accepted for publication: 1/22/01.

TMR Context

Over its existence, TMR has been implemented in over 40 different sites. It has been implemented in 14 different medical specialties. It has operated in outpatient settings, inpatient settings, intensive care settings, and combinations of these settings. The smallest application was a solo practitioner, and the largest site supported more than 350 providers. The maximum number of patients supported by TMR in one system was approximately 750,000. The Medical Record provides an integrated clinical and accounting functionality. At its peak, it was operational in 15 different sites. At present, TMR is operational in four sites. It is still a character-based system, uses DEC’s VMS operating system, and runs on the Alpha computer.

The Medical Record was developed in an environment with limited resources—we were pushing the
envelope of the technologies and the tools that were available to us. Over the past 30 years, the evolution of TMR has been driven primarily by the requirements of the myriad users. The advances in our work were a direct consequence of solving problems, and we were particularly fortunate to have selected a design architecture that provided freedom in evolving the system. We never faced a problem that we were ultimately unable to solve. In all our applications, we involved a clinical expert in the domain of implementation, and all our systems were tested in real-world environments, often removed from our physical setting.

**Development and Evolution of The Medical Record**

Our early work began with the development of a 19-page, mark-sense screening medical history. Around this time, the minicomputer was beginning to appear on the scene. The history-taking project was first initiated on a large mainframe but failed because of cost and the inability to get program, mark-sense output tape and the punched cards containing the free-text data together at the same time. By interfacing the reader directly to the minicomputer and entering the free-text data directly, we solved both the cost and timeliness problems.¹

The screening history was followed immediately by the development of a real-time, interactive headache questionnaire by Bill Stead (William W. Stead, MD, Vanderbilt University), a first-year medical student, working with the Division of Neurology. Patients completed the history directly on the minicomputer.² This project also provided our first experience with decision support, with the program identifying the type of headache. The diagnostic accuracy matched that of the neurologist. Unfortunately, the neurologist did not need a program for help, and the cost of the hardware was too great for primary care sites, which had a need but only a small number of relevant patients. Other history-taking programs developed over time included a pediatric well-baby questionnaire³ and work with an audio version of the screening history.⁴

In passing, I mention that we proposed a hospital information system for Duke Hospital based on a group of minicomputers connected together. We almost convinced Duke to make the commitment, but instead Duke supplied funding to begin the development of a computer-based medical record. We selected the Division of Obstetrics as the focus of that project, for several reasons. First, the Division was willing. Second, the obstetrics clinic was in a different location from the delivery suite, and records had to be in two places at once—difficult in paper form. Obstetrics also used a number of forms, and the people were accustomed to controlled data capture. Data were reasonably well defined, and the report requirements were known.

The obstetric medical record was developed on a Digital Computer Corporation PDP-12, shown in Figure 1. Each DEC tape would accommodate the records for only one month's pregnancies. Patients' records were stored on the tape by the month of expected delivery, and Bill Stead slept in the computer room to change the tapes as required as we crossed each month's boundary.

We computerized what was done manually. Although external “tables” were used to define display screens, the application was primarily hard coded. We began with mark-sense forms to collect the prenatal history and another mark-sense form for the obstetric examination. The output was in narrative form and often exceeded eight pages. We were not popular, and we quickly learned the definition of information overload.

---

¹ Figure 1: The PDP-12 minicomputer, manufactured by the Digital Equipment Corporation. This minicomputer had 4 KB of 12-bit words as main memory, a 20-character by 12-line cathode ray tube display screen, and a 300-KB DEC tape for secondary storage.
Fortunately, Bill Stead, as a second-year medical student now, was on obstetric rotation. He saw firsthand what was actually needed and overnight rewrote the output program to create Junior Resident admit note. Figure 2 shows the outline for an admit note taken from the 1971 student orientation manual and the computerized version of this note. He extracted the appropriate data from the previously scanned data and reformatted the output to match what had previously been written by hand, leaving space for the five additional items required but not in the database. The users were now happy, because the computer made life easier by meeting a need and reducing their task load. Part of this output was a half-page “starred box” that contained all the data of essential importance and included pertinent medical problems.5,6

We wrote these programs in assembly language simply because there were no other adequate languages available for interactive minicomputers. We had written several programs, each dedicated to a specific purpose—INPUT, FRAMES, MASSAGE, PRINT, RETRIEVE, and STORE. These programs were run in sequence. They worked but created an undesirable overhead; for example, if you needed to use MASSAGE to do a calculation in the middle of a branching questionnaire, you had to switch programs.

In 1972, we decided to formalize these tool-kit programs as a formal programming language that we called GEMISCH.7 The functionalities of the individual programs were converted into commands in the new language. The new language represented a considerable increase in the scale of what we could produce. We still use it today, and it has grown into a powerful, high-level database management. It is interpretive, translated into a pseudocode for execution.

GEMISCH is characterized by extremely powerful text manipulating functionality, smart print-generating capability, variable and flexible file types and file-handling routines, display functionality within the limitation of character-based and other features that made it easy to interface to any type of device or provide any functionality.8 I note that many other groups developed their own languages during this period, including MUMPS and HELP. In spite of the criticisms we have heard over the years, controlling our programming environment enabled us to survive four changes in operating systems, meet a variety of interface requirements and make the programs do whatever was necessary.9 With the addition of GUI
I am not convinced that continued use of this language is not in our best interest.

We next developed a computer-based record for primary care. This application used a different design and was transaction oriented. Each encounter was a record, and encounters were linked by the patient ID. Although we started this project strictly as a clinical CPR (computerized patient record), we quickly understood that for the system to be affordable and used, we must also meet the needs for practice management, such as scheduling and accounting. A new family medicine unit was formed, which was affiliated with the Durham County Hospital. Although this group started with the identical CPR system that was implemented at Duke, within a month the system had changed so much that applications were two different packages supported by two different groups. We quickly found ourselves supporting four sites, with a programming staff of 25 people. We needed a solution that would permit the same program to be used in all settings and reduce the support requirements.

By 1975, our concept of what would be required to support a CPR began to have form. Our experience had further taught us the limits of reproducing documents that looked like the ones that worked with paper-based procedures. Bill Stead was now a renal fellow, and he took one year—his only dedicated extended period of time—and worked with me.

During 1975, we took a new approach to the electronic patient record. Bill defined a data structure that represented data according to its meaning as contrasted to its planned use. He then prototyped modules to capture into this structure the records of the patients he was seeing in his nephrology clinic. I then programmed the modules needed to use this structure to support practice management aspects of the primary care clinics.

The resulting computer-based patient record—TMR—had the characteristics that we felt were significant and necessary for handling the multiple features required to support a variety of clinics:

- Modular construction which simplifies programming, documentation and maintenance
- Data-independent programming through data definition dictionaries, which permit the same program to function differently in a variety of clinics
- A combination of problem-oriented and time-oriented formats to generate a record that would more completely satisfy the need for patient care
- Two input modes—a parameter-oriented, direct input mode for source data entry and indirect input using paper or dictation to capture input for subsequent input by third party
- The direct coupling of protocols to data entry and data display

The design of the new TMR took about three to five years before we were able to enter a complete medical record into the computer. The renal clinic at the Durham VA Hospital provided our first experience with real-time data entry by providers. The physicians were required to enter all prescriptions directly into the computer. Figure 3 shows the rolling cart that provided access to the terminal used by physicians to input the prescription data. The high repeat rate of medications for this population of patients meant that drugs could be re-ordered with a minimum of keystrokes. This was clearly a time-saving activity and was well received by the physicians. The record included demographics, problem list, drugs and allergies, and laboratory data as well as history and physical examination findings. The physicians entered these data directly when it made sense. They used notes on encounter printouts when paper was a better interface. Figure 4 shows the paper renal encounter entry form...
on which data were entered by both nurse and physi-
cian for subsequent entry into the computer by a data
entry person. In these cases, a physician assistant
“brought the record up.”18

University Health Services Clinic, Family Medical
Clinic, and later obstetrics were converted to TMR.19
One program supported the four sites with differences
and customization accomplished through a metadata
dictionary. Working with physician-focused clinical
systems and patient-management- focused systems
was a valuable experience. We learned to tailor the
capture of data to the user. We learned to block delays
in performance to natural transition times. We learned
to maximize data stored on a single screen and to min-
imize keystrokes for entry of data. Figure 5 shows a car-
diology data entry screen. We designed frame driv-
ners that collected input data from a number of diag-
nostic tests, including cardiac catheterization, echo
cardiograms, and treadmills. A template, including
the most likely data to be entered, was defined in the
dictionary for each doctor. It also required new
report-writing capability to translate structured data
into narrative reports and an inverted file to optimize
cross-record retrieval.24,25 The TMR protocol system
was also used to track patients with coronary artery
disease for follow-up.26 Research needs also required
the development of a query system.27

In 1982, we also installed our first inpatient system in
the Kenneth Norris Cancer Research Hospital here in
Los Angeles. That site still runs TMR. As TMR was
installed in additional settings at Duke, we needed to
interact with the hospital information system.28
Kenneth Norris Hospital also was the site of the devel-
opment of an independent (but real-time interfaced
with TMR) laboratory system.29,30 This experience
taught us the differences between focused, departmen-
tal systems and the computerized record. We learned
what should be stored in the laboratory system for
management purposes and what needed to be passed
back to the CPR. We learned about synchronization of
databases and events, such as system backup, and we
learned about persistence of data in the patient’s record
and in the departmental system database.

We learned there are major differences between inpa-
tient and outpatient systems.31,32 The volume of data
increases significantly in the inpatient setting.
Patients stay for longer periods of time. Late charges
are a fact of life. Patients are grouped in wards rather
than from an appointment list. On the other hand,
decision support has a more immediate value. Drugs
are more powerful, and the risks of drug–drug inter-
actions and adverse reactions are greater. We defined

The Division of Cardiology had created a research
database for patients with coronary artery disease,
starting in the late 1960s.22 In 1983, that application
was moved to TMR and coupled with clinical func-
tionality. The question was, could data acquired as
part of the process of delivering care be used as a
research database? To a large extent, with interesting
exceptions, the answer was yes.23

Supporting Cardiology required additional function-
ality in the form of new data collection screens, new
displays, and new data types. Figure 6 shows a car-
diology data entry screen. We designed frame driv-
ers that collected input data from a number of diag-
nostic tests, including cardiac catheterization, echo
cardiograms, and treadmills. A template, including
the most likely data to be entered, was defined in the
dictionary for each doctor. It also required new
report-writing capability to translate structured data
into narrative reports and an inverted file to optimize
cross-record retrieval.24,25 The TMR protocol system
was also used to track patients with coronary artery
disease for follow-up.26 Research needs also required
the development of a query system.27

In 1980, a start-up family practice unit in Los Angeles
purchased TMR for installation into their clinic of 35
providers. This, our first non-Duke implementation,
was the first site to use all the pieces of TMR in one
setting. The academic setting had tended to use only
aspects they were pioneering. This experience taught
us the difference between a local academic setting
and a remotely supported private setting.20,21

Figure 3 View of rolling cart, used to move computer
terminal to hallway outside examining rooms in the
Durham VA hospital. Dr. Bill Yarger is shown entering
prescription data into the terminal for a patient he has just
seen. Physicians were required to enter prescriptions
directly into this terminal after a patient visit.
new types of displays and new pathways through the system. The inpatient system tracked patients as they moved from one location to another. Tests that had to be scheduled were queued for scheduling when they were ordered, resulting in considerable time savings and increased efficiencies. Table 1 compares what we learned about the outpatient, inpatient, and intensive care environments.

In 1988, we began the bedside pilot project in a surgical intensive care unit, in which we interfaced to a number of bedside instruments and implemented online nursing notes. Over the years, we continued to try to involve the providers of care in the electronic system. Key requirements were response time, clarity of use, minimal effort, and return on investment. The addition of decision support features and evaluation in efficiency and quality support the case for ROI. We developed a philosophy for the demand-oriented medical record—the ability to deliver what the provider wanted when and how they wanted it. Laboratory data were delivered to providers as an e-mail messages that could then be annotated and e-mailed to the patients. Automatic links to literature retrieval were provided, with the results linked into the patients’ records.
The obstetrics system has continued to evolve over the past 30 years. At Duke, the system expanded to include some inpatient activity. Clinical records for the newborn babies were automatically created and included important data about the mother’s pregnancy and the birth process. The obstetric database now includes two generations of pregnancy history and birth data. The obstetric system made a significant impact on fetal mortality and morbidity when a regional database was established to track prenatal experiences of pregnant women over a five-county area. The computer was used to ensure adequate prenatal care and to make sure prenatal data were available at the points of delivery. The 30 years’ accumulation of rich clinical data in obstetrics has resulted in several research studies.

The multiple implementations of TMR throughout Duke required us to learn how to deal with multiple...
databases that must interact.\textsuperscript{35} An example is the seamless transition from one appointment system to another. We had to deal with different vocabularies, synchronization of databases,\textsuperscript{56} scalability and accommodating growth,\textsuperscript{37} interfacing to other systems for analysis,\textsuperscript{38} and development of data interface standards.\textsuperscript{59–61} We gained valuable experience in the operation of several types of clinical information systems.\textsuperscript{52–65} We learned how to estimate the requirements for clinical information systems.\textsuperscript{66–68} The different applications of TMR permitted us to understand how to transition from a paper system to an electronic record system.\textsuperscript{69,70} We experimented with different approaches in the use of paper and automation.\textsuperscript{71–73}

Major active development in TMR continued until 1997.\textsuperscript{74} Work on computerized clinical guidelines was integrated into TMR.\textsuperscript{75,76} Maintenance programming continues. For example, early this year we accommodated the required changes for the new APC codes. The TMR data, which are very rich in clinical content, continue to be mined for knowledge.\textsuperscript{56,77} New versions of TMR have been prototyped,\textsuperscript{78} but the long-term future of TMR has yet to be determined.

\textbf{Lessons about Clinical Systems and Databases}

A key factor in the long-term survival and ubiquitous use of TMR was the adoption of a data model in which input, storage, and planned use were all independent. For example, historical data could be input from a mark-sense short form or a long form. Input was direct from a scanner and from frames through which textual data were entered. Storage was organized in a modular fashion, by category of data, independent of input source or mode. Output could be a short narrative report, a long narrative report, or just a listing of the abnormal findings.

We stored in the patient’s record only the data that varied from patient to patient. Data were mostly structured, and free text was used to enhance meaning, to capture the gestalt of the interaction, or to substitute for a term not defined in the metadata dictionary. Data elements were stored as pointers to categorical sections in the dictionary of metadata.

The metadata dictionary, in compiled form, is stored as a fixed-length, directly pointer-accessed file. Figure 7 identifies the content of the metadata dictionary. It defines the data elements for each section of the patient record—problems, procedures, studies, therapies, clinical findings, supplies, providers, places, accounting, etc. The metadata dictionary permits the use of one program that runs in a variety of settings and a variety of clinical specialties. The metadata dictionary contains all the business rules for any given implementation. The dictionary contains such things as data element definitions, vocabulary and external code sets, physical resources, data capture protocols, billing algorithms, decision-support rules, work flow, information flow, linkages, management information, report generation, drug–drug interactions, and resources. Any and all data elements stored in TMR and the rules for their use are stored in the metadata dictionary. Figure 8 shows the detail for a problem entry in the data dictionary.

Limited resources and large amounts of data demanded a very compact storage of data. We also

\begin{figure}[h]
\centering
\begin{tabular}{|c|c|}
\hline
\textbf{Dictionary of Metadata} & \textbf{Metadata Dictionary}
\hline
Data element definitions & Information flow
\hline
Vocabulary and external code sets & Linkages
\hline
Physical resources & Report-generation rules
\hline
Data capture protocols & Drug–drug interaction
\hline
Billing algorithms & People and places
\hline
Decision support rules & Security rules
\hline
Work flow rules & Access control
\hline
\end{tabular}
\caption{Types of data stored in the metadata dictionary.}
\end{figure}

\begin{figure}[h]
\centering
\begin{tabular}{|c|c|}
\hline
\textbf{Problem Dictionary Line Content} & \textbf{Problem Dictionary Line Content}
\hline
Code set (e.g., ICD-9) & Linkage to management plan
\hline
Problem name & Problem-oriented display specs
\hline
Problem type & Protocol linkage
\hline
Classification & Parent diagnosis
\hline
Disease category & Coded modifiers
\hline
Linkage to causal DX & Intervention linkage set
\hline
Linkage to manifestations & Outcome linkage set
\hline
Linkage to S&P & Evaluation linkage set
\hline
Linkage to related studies & Nursing plan linkage set
\hline
Linkage to medications
\hline
\end{tabular}
\caption{Details of metadata entry for a problem.}
\end{figure}
knew we wanted to permit variable-length free-text data as well as coded structured data. The architecture of the GEMISCH record is shown in Figure 9. Each data element, called a node, is represented by a three-bit code, or trit. Numbers are stored as characters. The record contains a large buffer (initially set to 4 KB, now set at 64 KB), in which the text is entered from one end and node pointer pairs from the other end, moving toward each other. Text is terminated with a special character. The node-pointer pairs are in sort order and permit direct access to the text. A deletion clears the trit and removes the node-pointer pair. Updates are in place if an update is the same length or shorter or is relocated with a first-fit algorithm. When the two sets meet, an automatic garbage collection is performed.

Data are stored as delimited strings containing pointers to the appropriate metadata line, a date-time stamp, and result. The result may be any of a number of data types, including coded data which, in turn, is defined in the metadata dictionary. Time-oriented data may be stored by repeating the date, time, and result strings or as a sparse matrix, with date-time stored only once as a column heading. Data fields are variable length and permit repetition of data elements with a repeat delimiter. Health Level 7 messages use a similar delimited syntax. Figure 10 illustrates the storage of data in the GEMISCH record.

Post-compositional data elements can be defined at the time of use under control of the metadata dictionary, which ensures that the combination makes sense. For example, phase, timing and location may define a heart murmur. Rather than pre-composing these terms in the dictionary by defining all possible combinations, TMR permits the dynamic definition at the time of use, by sequentially prompting for additional detail. One physician might enter murmurs and stop. Another might add that it is systolic, and a third might fully define it.

The Medical Record programs use a modular design in which we localize the things that deal with a set of data. The modules are monitor, registration, appointments, problems, studies, history and physical findings, therapies, encounters, accounting, and subroutines. The continued development of TMR over the past 25 years has followed a pattern of first programming a specific solution to a requirement as a separate program or conditional code. The patch is then generalized, with the business rules defined in the metadata dictionary. Finally, each module is reworked every three or four years, and the new functionality is integrated seamlessly into the module.

Modules are much like classes. Some of the routines are components that are reused. For example, TMR includes a completer subroutine that is reused for every data entry. The completer interfaces between the user program and the metadata dictionary—which also contains a synonym list for each data element.

The basic patient record in TMR contains all time-independent data and the latest sets of time-oriented data. The amount of time-oriented data is controlled by volume of data. Older data are stored in online archive files by class, with pointers to the data contained in the primary record. Any data element, regardless of the volume of data, is online and retrievable with a maximum of two disk reads.

Data are stored redundantly to optimize tasks and enhance response time. For example, the encounter storage includes the problems and procedures that are dealt with in the encounter, the resources used, studies ordered, medications dispensed, supplies, professional fees, and disposition. Encounter problems, subject to business rules defined in the dictionary, are also stored in a summary problem section. The dictionary defines parent–child linkages, and a more precise definition replaces a more general definition in the summary problem lists. The study data are also stored as time-oriented data. Therapy data (prescription drugs, immunizations, and allergies) are stored with prescribing data in a therapy class section.

Other types of files enhance performance. Each updated record is stored in a daily file as well as the total file. At the end of the day, the daily records are used to generate reports, create the day’s statistics, and provide real-time backup of active data. Accounting data are transferred and redundantly stored in a collection’s workstation. Accounting data are used to validate the integrity of the files before transferring the daily records to the backup total file. Other files track protocols and other events that need to be monitored. An inverted file supports across patient retrievals. This inverted file actually has better response characteristics than some commercial databases.

This data architecture has survived for more than 30 years. We also wrote our own record management routines for optimum utilization of disk space. Patient records that are on the order of a few hundred kilobytes expand to tens of megabytes in a relational database system.

Two other lessons were gleaned from our experiences. First, decision-support algorithms need to be integrated seamlessly into the normal flow of the work process. Second, the CPR is more than a repository for
data. Enhancements to workflow and automatic linkages to events and actions made the programs much more useful. Examples include automatically queuing ordered tests that must be scheduled and linking the appointment process into the disposition event.

**Implications for Medical Informatics**

All application developments in TMR have been done with a clinical partner in the specialty area. The clinical partner is active in the practice; often the fact that the co-developer is using the system has prevented embarrassing errors and much more sensible design.

The most critical partnership in the development of TMR was the relationship between Bill Stead and me. As a partnership, we performed as a medical informaticist using Bill's clinical expertise and my technical expertise. Part of that relationship was each teaching the other. It wasn’t just “this is how you should do it.” It was getting a level of understanding so that we each knew why we were doing it in a certain way.

The reason we could meet any requirement was that we controlled our tools at the leading edge—hardware and software. You can do whatever you have to do to enhance response, accommodate volume, or interface a device. Examples include interfacing the scanner to the computer (at a cost of $15) or modifying DEC’s operating system to a seven-user time operating system. Early obstetrics system, typing a character on a teletype located in the delivery room captured the computer to service the interrupt. Having our own language permitted us to accommodate new operating systems and new computers without having to change the application. We satisfied most of the Y2K problems by changing the language. Even independent systems can be interfaced.

The modular structure of TMR promoted rapid prototyping as stand-alone projects. Periodically, these stand-alone prototypes are incorporated back into the module. The file design permits scaling the system to handle any volume of data or number of records—supporting any number of computers, servers, or disks.

Significant advances in hardware and software capabilities occurred over the lifetime of TMR. The first systems were implemented on a Digital Equipment Corporation (DEC) Linc-8 and then PDP-12 minicomputer with 4 KB of 12-bit memory. The data were stored on 300-KB small magnetic tape units. The display was a 12-line, 20-column cathode ray tube. Our first printers were teletypes with uppercase only, printing on rolls of paper. From there, we evolved to

---

**Figure 9** The GEMISCH record structure. This object-oriented format permits maximum compression of data and rapid retrieval of individual data elements.

**Figure 10** Database structure for time-oriented data. The first line shows the codes in display order delimited by “|.” A search of this line quickly identified what items were present. The second line shows the date-time stamp for items present (column heading) Subsequent lines contain the actual data values.

---

```
10115|25|21|22|67|32|45|148|53|162|54|215
000600800|99235|1015|991501500|981891215
150|155|165
1120
31|313
14.3|14.8|15|31|16.1
```

---

**Table:**

| 027770 | 043002 | 020100 | 002040 | 002067 | 000000 | 000000 | 000700 |
| 000000 | 000000 | 000000 | 000020 | 000074 | 000000 | 000000 | 000000 |
| 011201 | 000100 | 100002 | 077710 | 000000 | 000000 | 000001 | 000000 |

3-bit node states
text buffer
node/text offset

---

**Figure 23**
drum printers, then to line printers, and now to laser printers. Operating systems became available that supported multiple users and sophisticated file-handling systems. Video terminals evolved from 12-line 20-character displays to 24-line 80-character and then full graphic displays with multiple fonts. Memory size increased from 4 to 8 to 64 to 256 KB and then to megabytes. Storage devices increased from 300 KB to 2.1 MB, through 80, 300, and 600 MB, and now to gigabyte storage. Connectivity moved from twisted pair to coax to fiber. We used RADs, line drivers, modems, and finally Internet.

Change in technology only moves the challenges; it does not eliminate them.

The author acknowledges the contributions of Bill Stead in preparing this talk. His passion for history and documentation helped recall the experiences throughout the years. The author also acknowledges some of the many people who contributed to the development of TMR and GEMISCH over the years. The original development team included Alton Brantley, Steve Feagin, Steven Lloyd, Bill Stead, and E. L. (Skip) Walter. Mark Straube joined the team early and is the keeper of GEMISCH; he continues to adopt and modify the language. Jim Collins, Edward Hammond III, Ruby Grewal, and many others have also made contributions.

In the application areas, contributors from Duke University have been, in obstetrics, Rick Jelovsek, Bob Brame, Marvin Hage and Kay Schlitz; in family medicine, Harvey Estes, Steve Gehlbach, John Hansen, Kim Yarnall, and Lloyd Michener; in nephrology, Bill Stead, Bill Yarger, and Jim Fitzwilliams. Other contributors from Duke have been David Pryor, Rob Califf, Pat Blunden, and others in cardiology; Joe Moylan, Kevin Fitzpatrick, and all the nurses in intensive care; Skip Burton in dermatology; Al Heyman in neurology; John Rice in rheumatology; and Bill Peters in bone marrow transplantation. Contributors outside Duke have included Marshall Berns, John Casagrande, John Daniels, Michael Hillier, Jon Sternburg, Lois Funderburk, Kay Hammond, and many others.

References

66. Jelovsek FR, Hammond WE, Deason BP. Use of TMR in a
73. Hammond WE. Implementing CPR systems; the pathway to the paperless medical record. 2nd Natl Conf Comput Med Rec. 1993:137–47.