

**Building information systems in health care
-A reference guide for health care decision-makers-**

Author:
Ozren Pestić
Chief of Informatics Department,
General Hospital Zadar
Croatia.
opestic@zadar.net

April, 2004

Preface

This manual aims to introduce the reader to the subject of medical informatics--specifically how to build information systems in various health care settings. Although some parts of the manual present detailed knowledge of specific applications tailored to technical specialists, the overall purpose of this manual is to provide those involved in the development of information systems with a framework that will assist in the planning and design process.

In chapters 2,3 and 4, where some elements of system analysis are explained, I used some parts of the book 'ORACLE Designer Handbook' by P. Koletzke and P. Dorsey, Oracle Press 1999.

Please note that for simplicity, not all examples used in this manual are taken specifically from the health care environment. Also, Oracle tools are mentioned in several parts of the text as an example, but there are a variety of other tools available for developing such complex information systems.

I am exceptionally grateful Mark Storey,Irina Ibraghimova and Duško Jagodić for their great help and very useful suggestions.

Table of Contents

Preface	II
Chapter 1: Introduction to information systems - a vision of the future	1
What is a health care information system?	1
Goal	2
Data, information and databases	3
<i>Quality of data</i>	3
<i>Database</i>	3
Standards	5
<i>Information data model (IDM)</i>	8
Computer-based Patient Record (CPR)	10
Possible technological solutions for information systems	12
<i>Technical solutions</i>	12
<i>The role of the information coordinator</i>	13
Chapter 2 : Strategy phase	15
Where to start from?	15
<i>What is the purpose of the Strategic phase?</i>	16
Deliverables	17
Strategy document	18
I. Executive Summary	19
<i>Business and Financial Sponsorship</i>	19
<i>Motivation</i>	20
<i>Solution</i>	20
<i>Cost-Benefit Analysis</i>	20
<i>Conclusion</i>	20
II. Legacy System Description	21
<i>History of the Current System</i>	21
<i>Description of the Current System</i>	21
III. Related Projects	21
IV. Business and Financial Sponsorship	22
<i>Political Environment</i>	22
<i>Conflict Resolution</i>	22
V. Motivation	23
VI. Project Scope	23
VII. Solution	23
<i>Proposed System</i>	23
<i>Strategy Entity Relationship Diagram (ERD)</i>	24
<i>Process Flow Diagram</i>	24
VIII. Cost-Benefit Analysis	25
<i>Benefits</i>	25
<i>Costs</i>	25
IX. Project Organization and Staffing	26
X. Workplan	27
XI. Business Impact	28
XII. Strategy Document Conclusion	29
<i>Application Partitioning</i>	29
<i>Scope</i>	29
Strategy Document Example	30
Modifications for Smaller Systems	34
Chapter 3 : Analysis phase	35
Deliverables	35
Overview of Information Gathering	35
<i>Eliciting Requirements from the Users</i>	36
<i>Interviews</i>	36
<i>Open-Ended Interviews</i>	37

<i>Questionnaires</i>	41
<i>Electronic Communications</i>	42
<i>Joint Application Development (JAD) Sessions</i>	42
<i>Modifications for Smaller Systems</i>	43
<i>When Is Information Gathering Complete?</i>	43
Chapter 4: Overview of the Test Phase and Implementation Phase	44
Testing Phase.....	44
<i>Test Plan</i>	44
<i>System Testing</i>	44
<i>Auditing Major CADM Deliverables</i>	45
<i>Performing Integration Tests</i>	46
<i>Performing Transaction Flow Tests</i>	46
<i>Performing Stress Tests</i>	46
<i>Validating Data Migration</i>	46
<i>Performing Backup and Recovery Tests</i>	46
<i>Evaluating Internal Controls</i>	47
<i>Handling Test Results</i>	47
<i>Determining Who Performs Testing</i>	47
<i>User Acceptance Tests</i>	47
<i>Performing small Pilot Lab Tests</i>	47
<i>Training and Documentation</i>	47
<i>When Is the Test Phase Complete?</i>	48
Implementation	49
<i>Overview of the Implementation Phase</i>	49
<i>Implementing the Entire System at Once</i>	49
<i>Phasing in the System</i>	49
<i>Handling Implementation Problems</i>	50
<i>Implementation Smaller Systems</i>	50
<i>When Is the Implementation Phase Complete?</i>	50
Versioning.....	51
<i>Version Testing</i>	51
<i>Modifications Necessitated by Software Product Changes</i>	51
<i>Maintenance Modifications for Smaller Systems</i>	51
<i>When Is the Maintenance Phase Complete?</i>	51
Conclusion	53
Chapter 5 : Experiences in developing IS in the world.....	54
<i>Building a national information system in the UK</i>	54
<i>The European Union (EU) approach</i>	54
<i>The Netherlands Leiden project (HIScom)</i>	55
<i>Diogene HIS in Geneve Switzerland</i>	55
<i>The Cleveland Clinic Foundation</i>	55
Experiences in Croatia.....	56
<i>Initiative of Ministry of Health</i>	56
<i>Experiences in the General Hospital of Zadar, Croatia. (GHZ)</i>	56

Abbreviations

CADM – Case Application Development Method
CASE – Computer Aided System Engineering
CPR – Computer-based patient record
DBA – Database Administrator
DBMS – DataBase Management System
DDL – Data Definition Language
DML – Data Manipulation Language
ER – Emergency Room
ERD – Entity Relationship Diagram
GP – General Practitioner
HCFA – Health Care Financing Administration
IS – Information System
JAD – Joint Application Development
JCAHO – Joint Commission on Accreditation of Healthcare Organizations
LAN – Local Area Network
LOS – Length of Stay
MR – Medical Record
OS – Operating System
RAC – Real Application Cluster
SQL – Structured Query Language
WAN – Wide Area Network
WHO – World Health Organization

Chapter 1: Introduction to information systems - a vision of the future

What is a health care information system?

A health care information system is a collection of programs and procedures, which enable data entry, storage, and the use of health care and administrative data about patients, healthcare workers, healthcare institutions, and other institutions connected with healthcare.

It consists of the following components:

- primary health care information system
- outpatient management information system
- hospital information system
- public health care information system
- pharmacies information system
- health care funding information system
- Information system of Ministry of health care
- information systems of various institutions connected to health care

This is often referred to as an integrated information system because it integrates all segments and processes in health care.

Guides that analyze approaches to health information systems frequently take a historical approach—looking back upon the successful and failed experiences of those who have tried to implement them. This guide instead uses a forward-looking approach, starting with a vision of what the integrated health care information system should look like.

Before we start identifying and analyzing this vision, however, it is important to emphasize that the purpose of building an information system (IS) is to improve health care. The goal, as defined by the World Health Organization, is to make health care available to all people, regardless of geographical location or socio-economic status. When we talk about IS, we think of an IS that will help the community to achieve that goal.

Scenario:

Situation 1:

The patient is at his home, has an Internet connection, and has access to his primary care physician (GP) and the emergency room (ER) of the local hospital. He can search for information about his illness and symptoms on the authorized web pages of the Ministry of Health or an accredited institution. There he can find information about new drugs and procedures related to his health problems. He can make an appointment for a visit to his GP, entering data and explaining the reason for his visit. In some cases (e.g., chronic disease) he can ask for advice from a specialist from the hospital. Furthermore, he can see the information about different teams and their success in the treatment of the problems from which he is suffering. On the Web site of the Health care fund he can also read about recent decisions of government agencies relating to patient rights.

Situation 2:

The patient visits his GP, and the physician, using the computer-based patient record, can observe the patient's health problems. On the basis of the patient's new symptoms, he sends the patient to the hospital for additional examinations. Access to knowledge databases and examples of good practice enable the GP to follow the most current practice in medicine. The results of the additional examinations are transferred to the hospital through the national information infrastructure. Departments in the hospital see this request and plan their work accordingly. This visit is statistically noted in a public health care database and by the health care insurance fund.

Situation 3:

The patient visits various diagnostic departments, undergoes tests, and after he has finished all of the examinations, the results are sent to his GP via computer. All results are in the hospital database. If these results are of importance for the public health, they are stored in the public health IS database. The patient visits his GP, who prescribes a therapy using reports from the hospital and computer-based patient records (CPR). A request for a drug prescription is sent to the pharmacy's database, and the patient collects his drugs. Data about the visit are sent to the health care insurance fund for billing. The national database for emergency cases is updated with data from this visit. If the patient has severe problems and is traveling away from home, any authorized doctor in the country can access this patient's data, which may be important in cases of emergency (for example, by providing critical information related to allergies or chronic disease).

Goal

What is the final goal? A 'single' comprehensive, eclectic, multiprovider, longitudinal lifelong health record. The health record should contain everything about a person's health and the care received regardless of the provider, time, place, insurance plan or health care delivery organization. The record must be confidential but available to and usable by individuals with a legitimate need to know. It should be a record that can be used for the multiple purposes required by our structured and complex delivery system, including statistical analysis, quality of care assessment, and clinical outcome research as well as personal care delivery. The record should accept input from multiple sources and devices and should improve efficiency and provider productivity. The record should also be linked with knowledge sources and decision support resources to assist the providers in improving care.

The process of building such a robust health care information system is a difficult task. Can it be done? Can we build this kind of health care information system? I believe it can be done. Though it will take a long time, I am still very optimistic.

If we agree that this concept is our goal, we must ask ourselves, what is the theoretical and practical foundation for building such a system and what methodology should be used. This first chapter should give some answers or at least a clue.

The examples above included several terms and expressions that are central to the concept of information systems development: database, national information infrastructure, computer network, computer-based medical record, national knowledge databases, good practice, and standards. The next section will provide a basic understanding of these concepts in order to create a broader picture of the topic of this manual.

Data, information and databases

Data are things known or assumed; facts or figures from which conclusions can be drawn. The term data is a plural term and is used whenever more than one data element is described. Datum describes single data elements. Data are the center of decision making in health care. Data should be reliable, complete and well structured.

Information is similar to data, but it is much wider. Information is derived from data.

Data are raw facts and figures that are yet to be processed. When the processing is completed the data become information.

Information is data that have been manipulated in some way to make them valuable to the user. For example during the ECG (electrocardiogram) test, voltages are measured over a period of time. Every point on the graph represents data, but only after whole graph is available to the physician can he interpret this data and process it into information: if someone's heart is functioning properly or not. Sometimes we also use the term «information» to refer to data that can be stored in the computer.

There are a lot of data categories; socioeconomic data for the patient, financial data, patient identification data, clinical data, aggregate data. Primary data are those obtained from the original data source, for example data from a patient record. Secondary data are data sets derived from primary data, for example morbidity of the population. Data sets support and encourage the uniform collection and reporting of data. We will discuss this in more depth when we discuss the problem of standards.

Quality of data

Characteristics of Data Quality

- correctness, validity
- reliability
- completeness – all required data are stored
- legibility – data are readable
- currency – data are recorded at or near the time of event
- timeliness – length of time is minimized between registering data and its availability
- meaning – understandable information
- accessibility – available to authorized people when and where it is needed

There are many individuals and groups who rely on health data and who demand quality in the data collected, analyzed, interpreted and reported: physicians and other clinicians, nursing staff, public health institutions, third party payers, attorneys and courts, quality assessment teams, researchers, administrators and others. Various countries and institutions have tried to solve the problem of data quality in various ways. One method is to train health information management professionals who are responsible for ensuring data quality. This might help, for example, in improving the way that data is coded. Most information systems are based upon codes, which means that if the incorrect code is used, the picture of health care data can be totally different.

There are a variety of methods to ensure the quality of data. Each institution must build its own methodology. One of the methods established on the national level is to connect accreditation policy with data quality. In the US, the Joint Commission on Accreditation of Healthcare Organizations (JCAHO) can refuse to give accreditation to hospitals where the quality of data is very poor. For example, if the number of delinquent medical records (MR) is greater than 2% of monthly discharges, the hospital receives a recommendation which must be resolved in order to retain accreditation.

Database

A database is a collection of data that is related to a particular purpose. A software shell around the data assists the user in storage and retrieval; it controls the access to the data and keeps a log file of all data transactions. This software shell is called a Database Management System (DBMS).

The typical structure is:

Database

File (data storage entity)

Record (smallest units of data storage)

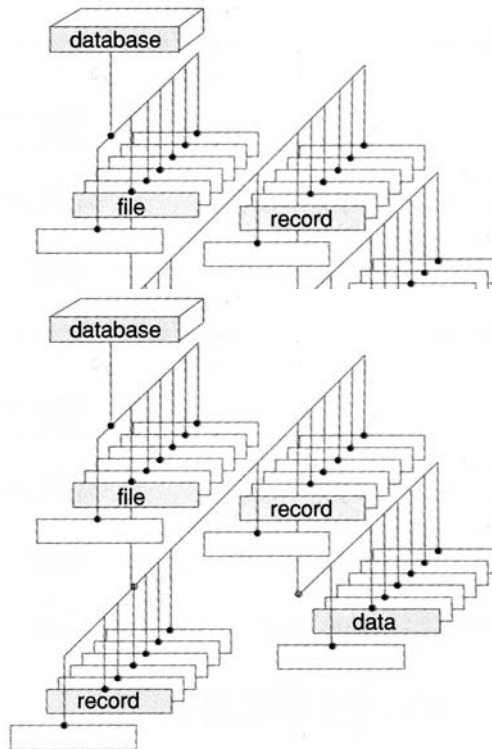


Figure 1. A database consists of a set of files managed by the DBMS.

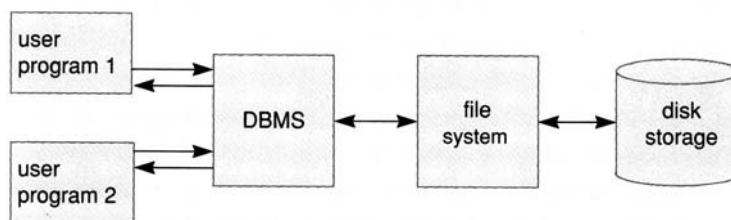


Figure 2. 'Position' of DBMS in the overall system

What are the main tasks of DBMS

- taking care of all database storage, modifications and retrieval operations
- checking data integrity and consistency rules
- access control
- multi-user access (concurrency control)
- facilities for data protection (transactional logging)

The relational data model, which is the most widely used, can be described as a series of tables, where rows of tables represent records and the column of tables represent fields. RDBMS is a database management system for relational databases. The language for data definition and data manipulation in relational databases is SQL – Structured Query Language.

SQL consists of:

- DDL – data definition language used for creating database objects (tables, indexes, etc.) and
- DML – data manipulation language used for manipulating data in database.

Relational DBMSs have gained popularity because of their performance and the simple underlying concepts and query languages that have made the development of applications much easier. In RDBMS, data storage and semantics (the underlying rules and procedures used for control of values and meaning of data) are separated; semantics can only be provided by application programs.

Object-oriented databases represent one of the more recent developments in database design. An object-oriented DBMS is designed to provide richer mechanisms both for specifying data semantics and for implementing the methods that perform database operations. In fact, a set of related records can be seen as one object. The methods for setting and updating fields, for preserving its integrity, and for doing special computations can all be associated with each object. This type of database is still developing, however, and is not as widely used as the relational RDBMSs.

Standards

A standard is a clearly defined and agreed upon convention for the operation and behavior of specific computing functions, formats, and processes.

What are the benefits of using standard?

- simplifying data exchange between systems
- enabling system users to gain better and easier access to data.

Definition of terms:

Thesaurus – is a list of terms used for a ceratain application area or domain. A thesaurus is always intended to be complete for its domain.

Controlled vocabulary – is a restricted set of preferred terms.

Taxonomy – is the science of classification.

Nosology – is the science of classification of diseases.

Nomenclature – is a systematic listing of names of terms.

Classification – is the grouping of similar items together.

Coding – is classifying data and assigning a representation for that data.

Standard nomenclatures and classifications

In developing a health care information system, agreeing upon a common terminology for medical conditions, treatments and procedures is a critical, yet often complicated step. Attempts have been made to name and classify diseases and operations for centuries, and many methods have been used. Hippocrates was known to have classified diseases according to the part of the body they affected. In the early 20th century nomenclatures were often developed for use in a particular hospital. Having hospital-specific nomenclatures, however, made it difficult to compare information on diseases across the country. The Standard Nomenclature of Diseases was developed to solve this problem in USA. During the late 1940s hospitals began to use the International Classification of Disease for coding revised by WHO for use through the world.

Since this time, many standards have been adopted around the world. Here is a list of the most well-known and commonly used classifications and nomenclatures with short explanations and references for further reading.

- ICD-9, ICD-9-CM, ICD-10
- SNOMED-RT
- LOINC
- CPT
- RCC
- ATC
- DRG
- MeSH
- UMLS
- HL7

ICD – International Classification of Diseases

- Classification of causes of disease, morbidity and mortality
- Content: Diseases, syndromes, etiologies and injuries
- Organization: WHO
- Versions in use: 9, 9-CM, 10
- information:
<http://www.who.int/whosis/icd10/index.html>
<http://www.cdc.gov/nchs/about/major/dvs/icd9des.htm>

SNOMED-RT – Systematized Nomenclature of Human and Veterinary medicine – Related Terms

- Encoding of clinical findings
- Content: comprehensive coverage of clinical content
- Organization: College of American Pathologists
- has several axes: finding/conclusion/assessment, body structure, morphologic alterations, procedures, biological functions, substances, living organisms, physical agents, specimens, occupations, social context. Related terms mean that there is an explicit relationship between concepts.
- information:
<http://www.snomed.org>

LOINC – Logical Observation Identifier Names and Codes

- Names for observations mainly for laboratory and other diagnostic departments
- Content: Lab procedures, clinical measurements
- Organization: LOINC committee
- information:
<http://www.regenstrief.org/loinc/loinc.htm>
<http://www.mcis.duke.edu/standards/termcode/loinc.htm>

CPT – Current Procedural Terminology

- Classification of medical procedures
- Content: procedures
- Organization: American Medical Association
- example: 24290 Amputation, arm through humerus, open, circular
- information:
<http://www.ama-assn.org/med-sci/cpt/coding.htm>

RCC – Read Clinical Codes

- Encoding of clinical data
- Organization: National Health Service UK
- College of American Pathologists and NHS have announced merging Read codes into SNOMED-RT.
- information:
<http://www.clinical-info.co.uk/ct3.htm>

ATC – Anatomic Therapeutic Chemical Code

- Developed for the systematic and hierarchical classification of drugs
- Organization: WHO
- information:
<http://www.who.org>

DRG – Diagnosis Related Groups

- Classification of patients in the hospital
- Based on factors that affect the costs of treatment and length of stay(LOS) in the hospital, such as severity of illness, complications and type of treatment
- The resulting classes are homogenous with respect to costs, and they are medically recognized. DRG may thus be used for budgeting and reimbursement.
- In use mainly in the US, but a lot of European countries adopt this classification for reimbursement of health care services.
- information:
<http://www.hcfa.gov/medicare>

MeSH – Medical Subject Headings

- Thesaurus of medical terms
- Used to index medical literature
- Developed and maintained by US National Library of Medicine(NLM)
- information:
<http://www.nlm.nih.gov>

UMLS – Unified Medical Language System

- Cross-index of medical vocabularies
- It is a metathesaurus (thesaurus of thesauruses)
- Developed and maintained by NLM
- information:
<http://www.nlm.nih.gov>

HL7 – Health Level 7

- Standard for health care data interchange
- important in multivendor applications
- Organization: HL7 (about 1500 members world wide)
- ANSI accredited
- information:
<http://www.hl7.org>

Information data model (IDM)

An information data model is a model that contains:

- common reference of terms and concepts
- information processes
- information flows

It is important to adopt a reliable information data model for collecting and processing common data. If we want to interchange data among primary health care centers, hospitals, and health care insurance funds, then common data, and a common data format, must be defined. It is desirable that an IDM be approved by consensus at the governmental level. If a consensus does not exist on the national level you will have to create your own IDM. It is a matter of experience and knowledge to predict all data that will be used in the future development of an IS. Many groups around the world are proposing that the Reference Information Model (RIM), defined as part of the HL7 standards, be adopted as the standard for IDMs. The European Union (EU) supports this proposal as the standard for integrating health care information systems in its member states.

Why do we need to know all this?

If we want to achieve our goal from the beginning, standards are to play an important role. If we want to exchange data, to send data from one application (e.g., a hospital) built in one programming language by one vendor to other applications (e.g., a public health institute) built in a different program and a vendor, without standards it would be impossible. The most important thing for one community or health care organization on the national level is to adopt standards. Without standards our idea of the computer-based medical record is nothing but fiction. Unfortunately many countries in transition have very poor standards, and it takes a long time to implement them, whether they adopt existing standards or develop their own. Health care institutions at the national level, the Ministry of health, medical academies, and various associations should initiate plans to adopt or develop standards. Without them, it is like making cars without roads.

In Croatia we have adopted or developed some standards, including ICD-10, ATC for drugs, a coding system for diagnostic and therapeutic procedures, and some data sets for registries in public health. However, we are still far away from establishing many other needed standards because adopting them takes time. For example, one team in the US has been translating SNOMED for Spanish speaking area for over 6 years. (See: Berra, C.M. : Trajectories: a Linguistic-based Method for the Development of the Spanish Version of SNOMED. Proc. AMIA Symp. 2000;:969.)

Computer based Patient Record (CPR)

Synonyms: electronic medical record, electronic patient record

Definition: A CPR is electronically maintained information about an individual's lifetime health status and health care.

CPR replaces the paper medical record as the primary source of information for health care, meeting all clinical, legal and administrative requirements. The CPR is not an object, or a product, but a set of systems that must interact to support clinical workflow and related processes.

Functionally, CPRs

- are not massive databases, but independent computer systems at individual care sites, which, with appropriate security, access specific data from any system.
- provide access to complete and accurate patient data, clinical reminders and alerts, decision support, and links to bodies of related data and knowledge-bases.
- facilitate the capture, storage processing communication, security, and presentation of health information.

Types of medical records:

- time-oriented MR (data are stored and sorted by time)
- source-oriented MR (data are stored and sorted by source of data)
- problem-oriented MR (data are stored and sorted using a specific structure – S for subjective, O for objective, observation, A for assessment and P for plan (SOAP))

Types of data entry:

- structured data entry
 - more control on data entry
 - more reliable and complete data
 - time consuming
- templates
 - predefined forms of data in which you can change the attributes of predefined observations and therapy.
- free text
 - natural language processing (NLP)
 - users like it (their own choice of words and amount of text)
 - indexing of text for text retrieval
 - use of thesauruses for parsing text
 - problems in semantics (e.g. cough confirmed or denied)
 - biggest problem of NLP is not what is written, but what is not. If physician does not write down the text, it is lost for information retrieval.

Perhaps the best form of data entry is a combination of all types. With structured data entry you will 'force' someone to enter data that are important, but in certain sections you can allow users to write using free text because this is more comfortable for them. The support of your 'customers' (all who work on data entry) is very important for the success in developing and implementing an information system. Don't assume you can force or persuade them with reason to do things they are uncomfortable with. More often than not, the problems will come back to haunt you and will jeopardize the entire project.

We can define subsets of a CPR to reflect different health care settings. For example, we can speak about a hospital CPR when referring to data in the CPR that are collected at the hospital. A general practitioner CPR includes data that are collected in an outpatient clinic. It is becoming increasingly

common now in some western countries to use nurse medical records, which enable nurses to organize the care of patients using their own documents and data.

More reasons to accept model of CPR.

- As health care expenditures grow, 28% of the costs go to administering data, and 6% of the costs stem from problems in manipulating health care information.
- Paper based record causes a lot of costs (according to HCFA):
 - A physician spends 38% of his or her time on paper work.
 - A nurse spends 50% of his or her time on paper work.
 - 30% of requests are lost.
 - In 11% of all cases, tests must be repeated because the original results were lost.
 - Errors due to illegible or incomplete results for consultants.
 - Privacy risks – no audit trail showing who is accessing MRs.
 - Physician decisions control about 85% of healthcare costs. If we consider paper MR as their only tool, a reduction of costs seems impossible.
- Information management is central to healthcare decision-making.
- CPR seems to be the right answer to these problems.

Where do we stand today?

Even in developed countries there is much work to be done in this field, even though the basic steps have been taken. There are a lot of organizations and institutions trying to define the framework and standards for building a CPR system. However, there still needs to be an agreement on many things, including:

- vocabulary – so that the meaning of terms can be trusted and information can be compared among institutions
- conventions – to be used to transmit data from one computer system to another
- communications protocols – if an electronic superhighway is to be achieved within the health care industry
- basic criteria for managing confidentiality and security within the information infrastructure – if the public is to accept the technology benefits inherent in patient data automation

It is important to organize institutions similar to the Computer based Patient Record Institute (CPRI – <http://www.cpri.org>) in the US through which we can gather together people and institutions that can contribute to the development of a CPR system.

Technology changes so fast--faster than our ability to adapt our health care systems to these changes.

It imposes new strategies in the education of health care professionals so that they will be able to use these new concepts and technologies in everyday work.

Possible technological solutions for information systems

The Internet provides a lot of new possibilities for building a communications infrastructure for a CPR system. Hardware is also not as complicated an issue as it once was; many components of a CPR system can run on both low-end and high-end hardware platforms.

The biggest issue facing IS planners now is the knowledge of how to build and implement an IS. This encompasses the following questions:

- what kind of technology to use
- what operating system(s) to apply
- what database to use
- how to organize LAN
- how to establish security on LAN
- how to establish security on Internet
- how to battle viruses

Determining the appropriate answers to these questions requires very specific knowledge. If staff at your institution do not already possess this knowledge, you must usually pay for these services, which can be very expensive. One day of a specialist's time can cost about as much as \$1000. Experience among institutions around the world shows that about 40 percent of the investment in information systems typically goes toward purchasing hardware, whereas the rest is spent on hardware maintenance, software, licenses, and consultation services provided by specialists. (See: Oracle magazine Nov/Dec 2001 p.101.)

There are various solutions depending on the situation in your community. If there is no governmental plan for IS in health care, you are on your own. It will depend on your ability to present, plan, and build an IS for your institution. It can cost a lot of money because you have to buy your own equipment and hire your own people for maintaining the system. It is not always easy to outsource all the tasks. Health care ISs are very complex, and users want solutions very fast. Sometimes you will not get support as fast as you need it. With your own staff dedicated to this task, you can develop your IS on a timeframe that meets your needs. However, sometimes it is hard to employ and retain qualified people, probably because salaries for high tech staff are generally lower at health care institutions and the best employees frequently leave to go to companies where they can earn more.

Technical solutions

- client – server

This solution means that the application software is running on each individual workstation (clients) and data are stored on a database server. . The main problem with the client-server model, which was dominant until a few years ago, is that it requires a lot of maintenance since every PC requires its own operating system and application software. This can become burdensome when your system has 300 PCs—or even 30. If a PC needs to be added or replaced on the network, you can connect a new one, but you must also install all application software and configure your computer for both the applications and the network. If your developer changes a piece of the program code in the application software, this change must be patched on all 300 PCs. These examples illustrate the amount of time that will be required for your IT team to maintain the IS. Thus, the client-server model will probably be useful only for small companies.

- web (intranet) solution

This solution assumes a three-tiered architecture: client – application server – database server. All application software is located on an application server, and each user on the network accesses applications using a browser. Maintenance is easier compared to the traditional client-server model because all changes can be made on the application server. On

the client side there are two solutions: a PC with an operating system or an autonomously functioning, or so-called "thin client," a PC which boots from the server. For a thin client, you don't have to install anything except for a piece of built-in operating system. Experience has shown this three-tiered model to be a cheaper solution because the less hardware involved, the fewer maintenance problems there are. While this solution does require that you buy a more powerful server, changes in hardware and software can be done on a small number of machines, so it is not time consuming and it is physically easier. This solution also represents the first step in making your information transferable and accessible through the Internet, which supporting the integration of your medical records with your IS. This could allow you to enable users—for example, GPs and staff from the Ministry of Health or from healthcare financing institutions—to access your data over the Internet. In Croatia, for example, which has a lot of islands, access to the nearest hospital is often difficult; there may be a ferry only once a day. Presently, the patient must come to the hospital for tests. If the test results are not ready during his visit, he must come back to the island the next day to collect them and then take them to his GP. But using the IS model described above, his GP will simply be able to download the patient's test results using the Internet. It is much easier and cheaper for the patient. By organizing our IS in this way, we have made the first step towards the integration of data into one system. But we are still far from developing a true CPR.

- regional data center solution

This solution is possible only when there is a plan developed at the governmental level. The most important assumption for this model is the existence of a national-level information infrastructure, which connects all health care institutions to a 'governmental intranet.'. The general idea is to establish a single datacenter that all institutions and users can access using this intranet. This model applies best in situations where each region includes several hospitals. For example, the region of Zadar in Croatia has three hospitals, and the larger region of Dalmatia has eight. With a regional datacenter, each hospital needs only a technician who will maintain the equipment and eventually one person for coordination. There is no need for a database administrator, an OS specialist, a network specialist, or operators for backup of data. We don't need servers or special rooms for it. We have estimated that for 80 hospitals in Croatia we would not be able to hire enough specialists to maintain an IS if such a system had to be installed and maintained in each hospital.

Then if you are on your own, there is one more important decision. Are you going to develop an IS with your own development team or will you hire a company from the outside (outsourcing)? These are different situations and you must explore and evaluate your own possibilities.

The role of an information coordinator

Suppose there is a person in charge of building an IS in the institution. Let us call her/him – the information coordinator. What is role of an information coordinator in all this? The answer depends on many things.

1. Maybe there are institutions where the information coordinator is the only person in charge of handling information technology issues. If this is case, an information coordinator's knowledge about all this may be important in the presentation of ideas and goals which should be achieved in building an IS. The information coordinator must know what possibilities exist for building such a complex system and what he or she can suggest to the chief physician or board. He or she also has to know if you want to build the system little by little in case there is not enough money for building it all at once. For example, you can start with admission and discharge of patients. Building an IS little by little is a possible solution, but you must always bear the whole picture in mind. If you don't, you will find after some time that you have built a 'tower of Babel.' You will have departmental systems built with various tools and various developers, and these systems may not be compatible with each other. If you don't tell developers what type of IS you would like to have in your institution, you could end up with systems that can not communicate with each

other. The info coordinator must be the person who can and will formulate the basic idea and strategy. This manual can help him or her to see what issues need to be considered.

2. In institutions where an information infrastructure already exists—where there are PCs, a LAN, and staff that maintains an IS, the role of the information coordinator is different. He is part of a larger team that will articulate the problems of building or rebuilding an IS. His or her knowledge about modern IS, CPRs, and standards will help in discussions about the methodology and goals of a new IS.

The information coordinator's tasks in building an IS and how he or she can participate in IS development teams are explained in the following chapters.

Chapter 2 : Strategy phase

A poorly planned project will take five times as long as anticipated. A well-planned project will take only three times as long as anticipated.

Where to start?

In the previous chapter, we talked about the basics, what a healthcare IS is and the basic principles on which such a system is built.

In this chapter we will try to give some directives about how to start with work on the planning and realization of the integrated information system.

Should you start developing a system with your own development team, or should you contract this job with an IT company - the principle of 'key in hand' (outsourcing)?

This is the question you should answer first. For an answer, careful analysis should be done, and it should be determined who will make the final decisions on this matter.

Experience suggests that it is wise to create a committee or task force that will analyze all situations, needs, and perspectives; make decisions and suggestions about what should be done; and then present these recommendations to the board of the institution. This could be called, for example, the IS Implementing Committee . The committee should include representatives from all staff profiles who will be using the IS, including:

- physicians
- nurses
- administrators
- members of the board
- IT specialists

These members should be:

- educated in basic knowledge of information technology
- respected by others
- people with a vision of development
- people who will be using the IS.

As it is easier to expose your ideas in a smaller group, the members of the committee will "defend" their decisions and suggestions in various situations. They will explain ideas to their colleagues and help them to understand the need for implementing an IS. In this way the idea and concept will start "living" before its implementation. With other committee members on board, you as an information coordinator will not be in a situation where the burden is only on you.

The question of implementing the information system is primarily a health care question, so it is important to gather representatives from all professions to implement it.

The decision whether to organize this committee and about its structure should be made by the chief physician and board, because the committee is serving as an advisor to the board and the chief physician.

The info coordinator or the person who will be in charge of the committee will need to prepare all relevant materials for this project. These materials consist of:

- identification of the need for informatisation
- goals and time-limits
- identification of who will develop a detailed project
- budgeting for the project

After these materials are written, the committee can suggest the policy for building an IS.

Who will develop the project depends on who will develop the IS. This might be:

- the staff of your institution
- an IT company together with the staff of your institution
- an IT company alone

If the institution engages an IT company, somebody should decide on how the selection between companies will be made and what kind of contract will be signed.

The next step is to form consultant teams from your institution (physicians, nurses, administrators) who should help in formulating user needs in the information system. They could help in solving the problems which arise from every day work and practice.

What does the project documentation consists of?

How do you start making documentation?

The first phase is always called the Strategic phase. A very common question in developing teams is: Why do we have to do a strategic document? Can't we just start coding? Maybe the next sentences will provide you with some answers.

What is the purpose of the Strategic phase?

The purpose of the Strategy phase is to formulate a basic description of the overall scope of the project and how the project will proceed. It is a "contract" between the ultimate users of the system and the people who will analyze, design and build it.

Spending an adequate amount of time on the strategy portion of the project is critical. Ideally, you should spend as much time as management will allow. A better understanding of what the project is committed to deliver will reduce confusion and help keep the project on course.

The more that is known about the underlying business goals and requirements early on, the less likely it is that there will be surprises during the Analysis phase or that the scope of the project will change dramatically. It is not necessary that every specific system requirement be reported at this phase, but all necessary information should be gathered.

On some projects problems arise when different individuals on the project team and within the organization think about the project in different ways. Development team after development team may come and go if they do not obtain a consensus of understanding among individuals involved at all levels as to the vision and the goals of the project. There are many talented technical consultants who can write code and programs galore, but without a solid overall plan and goal, these efforts are wasted.

There is a tendency to look upon strategic planning and methodology as an easy place to reduce costs. In the long run, this is a huge mistake and usually greatly increases the length of time for the project completion.

The first task of the Strategy phase is to develop a strategy plan. This plan will help answer questions such as the following:

- What will strategy document contain?
- Who will be interviewed?
- What committees will be involved in project?
- What will the deliverables be?

- How long will the entire system project take?

Several weeks may be necessary to produce an effective strategy plan. The plan for the Strategy Document should include criteria for determining when the Strategy phase is finished, what the expected deliverables are, and who ultimately signs off on the strategy part of the project before it proceeds to the next phase.

One useful aspect of the Strategy phase from both the user and development team perspective is that because it is impossible to accurately estimate the costs and resources necessary for the completion of the project until the end of Strategy phase, the users can contract with development team to complete work in this phase without committing to the full implementation of the project.

Deliverables

There are three deliverables for the strategy phase:

1. Preliminary Project Plan

This plan lays out the main tasks, deliverables, and the schedule for the remainder of the project. This project plan will evolve over the life of the project. In this plan, the resource requirements from both the user side and development team side should be listed along with the incidental costs for travel, hardware, networking and administrative support, and project administration. It is good to emphasize what types of events will influence the project and try to predict how it will influence the project. For example, how will the schedule be affected if hardware is not delivered on time? It is good to identify people and their responsibilities inside the project from both the user and development team side. In that way, the number of people who will need to be involved can be estimated.

2. Project contract

In the consulting environment, the project contract is frequently developed as part of the Strategy phase. In some cases it is finalized before the strategy phase begins.

Types of contract:

- fixed-price

indicates specified amount of money for which the system will be delivered. This type of contract has a very low probability of success. Users are not motivated to make the job easier for developers and developers will make only necessary things to fulfill the terms of contract.

- time and materials

Development resources are acquired based on the cost of hours worked and materials used. Users are highly motivated to finish the project as soon as possible, but developers are not motivated to complete the project. This type of contract can work only if users take a very active (approaching managerial) role in the project. All the risk is assumed by the user.

- hybrid

this contract combines elements of the previous ones. One example of a hybrid contract is time and materials contract with some percentage of billings withheld until project completion.

At the outset of the project, you need to identify what the conflict resolution mechanism will be when disputes arise. The rules of how such disputes will be resolved should be spelled out in advance of any significant work being undertaken.

3. Strategy document

This is the main deliverable. It is a high-level, focused description of the proposed system. It acts as a contract between the development team and the user organization. It should precisely describe the scope of the proposed system and, in broad terms, lay out the plan of how to accomplish the project. We will examine this deliverable in detail.

Strategy document

The Strategy Document should be as complete and detailed as possible to ensure that both users and developers have a clear understanding of what is expected.

Gathering information for the Strategy phase is done mostly through interviews. Notes from these interviews should be kept and can be referred to by a footnote in the Strategy document.

Goals of the Strategy Document

1. To communicate to management and all people involved in the project just what the project entails.
2. To act as sales tool for the development team. The Strategy Document proves the need for and importance of the project, clearly explaining why the project is cost-beneficial and efficient and demonstrating that the proposed plan makes sense, is feasible, and is the best plan to get the job done.
3. To serve as a contract between the development team and users.
4. To assess the scope of the project, list the promised deliverables, lay out the assumptions and limitations of the project, and limit the liability of the development team to what is agreed upon.
5. To provide a baseline for the proposed system, specifying the rights and obligations of both analysts and users, including the rights of the analysts to rethink the project when some things are altered. The document should make it clear that any major changes will affect the deliverable dates.
6. To provide a reference point and mechanism for handling conflicts that may arise.

Structure of Strategy Document

The Strategy Document should include following sections:

1. Executive summary
2. Legacy system description: history, current system
3. Related projects
4. Business and financial sponsorship
5. Motivation
6. Project scope
7. Solution: system, ERD, process flow
8. Cost-benefit analysis
9. Project organization and staffing
10. Workplan
11. Business impact
12. Conclusion

We will discuss each of them on the following pages.

I. Executive Summary

It is not just an introduction to the Strategy Document. It is a self-contained document and should be written to be read on its own. The executive summary represents the project team's current understanding of the total project.

"Consider the following anecdote about a study by the National Science Foundation of research grant proposal applications in US. These proposals reflect important research projects and are typically hundreds of pages long. Millions of dollars in grant money are distributed based on decisions made regarding the worthiness of the proposals of these documents. The study found out that the average amount of time spent on making a decision on an application was 15 minutes."(6)

The point of this story for a systems environment is that much of what is written is not read carefully, if at all. How much time is a busy executive realistically going to spend reading a document? The Executive Summary should convince him/her that they want to support the project as outlined, whether the costs are a few thousand or a few hundred thousand dollars. The Executive summary defines the scope, solution, development method, costs and benefits.

How long should this document be? Five- to ten-pages.

Probably, the final version of the Executive Summary will be completed after the rest of the Strategic Document is finished.

Business and Financial Sponsorship

Every project of any magnitude needs to have the appropriate commitment from the organization to accomplish its goals. The existing environment within an organization strongly influences how the project phases (especially the Analysis phase) take place. The project team needs to carefully document the following:

- who the players are
- the relative stakes in the project of each player and department
- the individual needs and requirements
- the mechanism of conflict resolution, including who has the ultimate authority

Motivation

The motivation section should describe the underlying business need for the proposed project. The motivation section must present a clear, concise, direct, and compelling statement of the user's reasons for wanting a system developed.

Here is an example of the motivation section directed to the management team of a large financial company that was building a new trading system:

"The existing system is an antiquated COBOL system held together by patches. It is only matter of time until there is catastrophic program failure causing an interruption in business. New products cannot be added to the system, which is preventing us from entering new markets. The existing system is incapable of expanding into new geographic regions. Management is considering 24-hour trading. The old system cannot support this at all. The longer we wait, the more this will cost our company."(6)

Solution

It should briefly describe what analysts intend to build, the basic vision of the overall system architecture, and how the system will work. It should summarize the solution section, described later in this chapter.

Cost-Benefit Analysis

The benefit versus costs section is devoted to estimating the costs of the project and the resources required to complete the project. It is important for the developing team to have a clear and accurate picture of resources that the user will have to provide and to present this in the document. The best advice is to be as realistic as possible.

Conclusion

End the executive summary with a few concluding sentences, not with numbers. Expectations for a formal business document include a conclusion. Part of the goal of the executive summary document is to communicate to the user that the development team is well organized and can be trusted to do the job completely.

II. Legacy System Description

This section describes the legacy system, including both the history of the legacy system and its current status.

History of the Current System

It is important to understand how the existing system was created. Without a thorough understanding of the legacy system, it is not possible to redesign the system. It is also important to find out about previous attempts at system redesign, and, if possible, talk to people who were involved in the original system design. In this way you can obtain important information regarding the legacy system. Include a walkthrough of the existing system and its development history in this section. The likelihood is high that this is not the first attempt to fix a system problem or design a new system. To avoid repeating past mistakes, examine the remains of earlier projects and any existing half-built structures. Documentation and analyses of these previous failures may help pinpoint the reasons for failures and help you avoid the same pitfalls. This information is well worth the time needed to gather it and incorporate it into the Strategy Document.

"Some individuals in the business reengineering community might argue that a thorough review of the legacy system is unnecessary and would hamper the completion of a quality reengineering effort. The fallacy of this point of view stems from the fact that, particularly with older systems, system requirements will be embedded in the legacy system that cannot be discovered any other way except by thoroughly reviewing the system. Treat the legacy system as merely another source of requirements. To ignore this potentially valuable resource is a mistake."(6)

Description of the Current System

"A new system can't be intelligently designed without a thorough description of the system it is destined to replace. Obtaining this information will require talking to more than one person. The analyst needs to understand more than just the functional aspects of the system. He or she needs to observe the system in use to get a sense of the supporting data structures (for a computer system), inputs, outputs, how the system is used, who uses it, what they use it for, the business reason for the system, and so on. Without a baseline measure, you will not be able to discuss benefits of the new system."(6)

This section should include an overview of the existing hardware, software, and network. Frequently neglected aspects of the legacy system are the reports it generates. Often, even after reengineering, reports may be similar to those provided by the legacy system. The Strategy Document should list the numbers and types of reports and give examples. These reports will be completely explored in the Analysis phase.

III. Related Projects

With rare exceptions, projects do not exist in isolation. The following questions must be considered in the Strategy phase of any project:

- How is the system that is being developed going to interface with existing systems?
- How and when will this integration be accomplished?

It is critical to consider how the integration with existing systems will affect the amount of time needed for the new system development.

"Sometimes, particularly in large systems projects, multiple teams will be working on different portions of development. In these cases, clear delineation of scope, interfacing, and deliverables is critical to project success. Trying to coordinate multiple teams on a project is exceedingly difficult and usually results in finger pointing and infighting when something goes wrong. This usually causes the project to fail. Having multiple teams, particularly from different areas, should be avoided whenever possible. However, having one team subcontract another team to fill in expertise where it is needed is acceptable."(6)

IV. Business and Financial Sponsorship

It is not politically correct to have a section in the Strategy Document called "political environment". Here "politics" is considered as factional scheming for power and status within a group or organization. However, from the developer's perspective, the political environment of the organization needs to be taken into account and documented. A complete description of the existing political environment is a critical piece of the Strategy Document.

Political Environment

"The political situation within an organization will influence the way any system is built. The political environment is just as important as the fundamental strategy requirements for the system. Just like changes in the scope of an ongoing project, a change in the political environment can cause the entire project to be rethought from the ground up.

Examples of changes in the political environment that can affect a project include personnel changes, department head changes, and the addition or elimination of a department involved in the project. A new person in a position of authority may have an entirely different agenda than his or her predecessor. A substantive upset in the balance of power among the players or departments can also cause a fundamental shift in project goals. Unless a way to handle these changes from the project perspective is carefully spelled out, the whole project may be in jeopardy should a change in the political environment occur. For example, if a main department head changes mid-project, the entire project may be pulled off track, and it may not be until another major organizational change occurs that the project gets back on track.

The best way to collect the necessary information on the political environment is to ask questions. Find out who the players are whose needs should be met. Define the organizational areas that the system affects. Create an organizational chart that includes all the people involved and goes high enough in the organization for its various branches to meet at a central person with ultimate decision-making power."(6)

The deliverables in this section include the following:

- Organizational chart
- List of the names, titles, and relationships of players
- Strategy-level requirements for each of the players or a description of how the needs of various players differ (each section may need to be developed with each player using a problem/definition format)
- Conflict resolution structure

Conflict Resolution

The approval process needs to be carefully spelled out and the following questions answered:

- How will it be determined when a particular deliverable is finished?
- How will changes in scope or deliverables be handled?
- What will be the sign-off process for each phase of the project?

When there are differences of opinion in spite of these questions, there must be a process for resolving the conflict. Usually, the best way to resolve conflicts on a project is to hold a joint application development (JAD) session. At this meeting, all of the parties are brought together to lay out all of the pertinent issues and forge a consensus. If a consensus cannot be reached, there should be a clear mechanism for a higher authority to arbitrate the final decision. All those involved should have agreed to this process before proceeding. This is another example of the importance of regarding the Strategy Document as a contract between the user and the analyst. These issues are often not stated explicitly but are truly salient and material portions of the contract. Everyone involved needs to understand and buy into the process as a whole in order for the project to be successfully completed.

V. Motivation

"The motivation section justifies the need for the project. The information you gathered in your interviews should include all the evidence you need. However, like a good journalist, a good systems analyst does not depend upon one source. The ultimate authority is the person with the overall responsibility for the project and the person or persons to whom the development team reports. The line of authority should be clear. If others present differing opinions regarding the project, it is the responsibility of the analyst to dig further and report these in the Strategy Document. It is important to keep the user informed at all stages of the project."(6)

VI. Project Scope

Even though scope issues will enter into other parts of the Strategy Document, scope is such an important topic that it should be dealt with specifically. Scope can be limited by any number of factors:

- Access to users: In one project, the scope was limited by whom the development team was allowed to interview. That limitation of access restricted the quality of the analysis and therefore also restricted what the development team could deliver.
- Subject area: One portion of a system may need to be replaced without making modifications to other portions of an existing system.
- Interface with existing systems: This interface can restrict scope. For example, building an inventory management system is one project. Hooking that system into an existing purchase order (PO) system is a separate project.
- Technical limitations: Developers may supply database design and application resources but not technical hardware, networking, or custom-written Windows DLLs.
- Role of the developers: This can be restricted to analysis and development and may not include training or user documentation.

Negotiation of scope depends upon the needs of the users as well as the capabilities of the development team. It is important that the scope of a project is specifically laid out, including what is within the scope and, often more importantly, a declaration of what is out of scope and will not be delivered.

VII. Solution

The solution section should summarize all aspects of the system to be built and include a Strategy ERD and process flow diagrams.

Proposed System

The discussion of the proposed system should lay out, specifically and in depth, what the development team will do for the user. This is not the place to discuss costs. This section should discuss the following:

- Hardware requirements - What equipment will be required to complete the project? Is a client/server environment to be assumed? Can existing hardware be reused or must new equipment be procured?
- Software requirements - What software will be needed? What licenses will be needed? Does existing software need to be upgraded?
- Deliverables List - The deliverables that are planned: for example, applications, reports, documentation, training, maintenance agreements.
- Knowledge transfer - This is relevant when an outside consulting team does the development. An important deliverable is that the consulting team provide a knowledge transfer to the in-house developers that will enable them to perform maintenance and enhancements throughout the lifetime of the new system.

Strategy Entity Relationship Diagram (ERD)

Depending upon the complexity of the business area and the level of knowledge of the project team and users, it may or may not be appropriate to include a Strategy ERD as part of the Strategy Document. However, this diagram should be created anyway, if only for the purpose of helping to focus the analysts' thinking.

"At the Strategy level, the ERD should identify the key entities and their relationships to provide an overall perspective of the business area data. For example, for a purchase order, the key entities might be vendors, approved items, purchase order, customer, and so on. This is not the place to be overly concerned about whether an entity is a lookup table or to define the cardinality of relationships. In addition to the diagram itself, the Strategy ERD must include narrative definitions of each entity. These definitions should be clear and concise. Errors in ERDs frequently occur because the analyst does not have a clear understanding of the entities being modeled. These definitions should come from the business users.

Entities in an ERD are not merely descriptions. They represent items of particular significance and interest to an organization about which they need to keep information. Demographic information, for example, is not an entity. The naming of an entity is a key factor in the success of an ERD. Good entity names facilitate communicating the business area information requirements. Critical to the success of an ERD are precise and accurate entity definitions. This is an iterative process, which may take some time. Good definitions of the relationships between entities are important because they reflect business rules."(6)

The basic process for creating an ERD is as follows:

1. Identify and name the major entities. At this point, the name is only relevant from a communications perspective. It is pointless to disagree over names before clear, concise definitions are written.
2. Define relationships among entities.
3. Write the best possible entity definitions.
4. Write the best possible relationship definitions.
5. Finalize the entity names.

From these steps, often the best entity names will emerge.

The entity definition always represents something in the real world. These definitions are crucial to the accuracy of the diagram. Entity descriptions should never start with "information on . . ." nor should they describe the attributes of the entity. The entity must be a noun. For example, the entity description of an employee could be "a warm, breathing body that works here." Each instance of that entity (row in the table) therefore represents "a warm, breathing body that works here." When entities are dependent, the description should include a reference to the parent entity. For example, a description of a purchase order detail might be "the quantity of a specific item purchased using a specific purchase order." Entities should always be discussed as things that exist in the real world. A problematic aspect of many ERDs is the naming of intersection entities. For example, the entities "patient" and "ambulance" have a many-to-many relationship. To handle this relationship, create an artificial intersection entity called "enrolment." The definition of enrolment is "the act of a specific patient visiting a specific ambulance." Therefore, each row in the table represents one specific patient visiting one specific ambulance.

To help others understand the ERD, generate tables and include sample data to clarify the way information is reflected in the diagram structure. The Strategy ERD should not have many entities; Strategy ERDs should be limited to 10-20 entities.

The diagram should fit on one A4 page. If the project consists of multiple modules, each should have its own ERD broken down by subject area. It is acceptable to have the same entity reflected on multiple diagrams.

Process Flow Diagram

"A process flow is a diagram that shows business processes and their interactions. In the Strategy phase, process flows are used to demonstrate to the user that the development team fully understands the business processes within the scope of the project. At this stage, process flows do

not need to be overly detailed unless more detail is needed to satisfy the user. Most of the work of creating a process flow is figuring out how the business processes interact and fit together, which usually takes place in the Analysis phase.

A strategy-level process flow can still be used to communicate the scope of the project. For example, for a purchase order (PO) system, the strategy level process flow will show what is being included and what is not. Will the system track the distribution of goods by matching the distribution against the PO when the goods arrive? Is this tracking part of another system? Does the system include the PO authorization process? These questions can be answered by looking at the process flow diagram for the proposed PO system. If an existing system is changed significantly, it is usually appropriate to deliver before-and-after process flow diagrams.”(6)

VIII. Cost-Benefit Analysis

The Strategy Document needs to show that the benefits of the proposed system will outweigh its costs.

Benefits

How can benefits be quantified? For a reengineering project, the cost-benefit analysis is fairly straightforward. For example, before reengineering, a report may take 14 person days to produce, and after reengineering, the same report may take only one minute to produce. Such a benefit is easy to document, and the advantages of time saved are obvious.

How can a value be placed on new flexibility within a system? One can look at the number of new reports generated by the legacy system in the last six months. If the legacy system took 10 to 12 person days to do what the new system can do in one or two days, then improvements in productivity can be realistically estimated and reflected as benefits of the new system.

Three types of benefits can be examined:

- Process improvements : time saved in completing tasks
- System modification efficiencies: Changes to the system, improving performance, new reports, and so on
- Ethereal improvements : executive information systems, ability to expand product line or market segment, ad hoc query tools, data warehouse, and so on

How can a value be placed on these types of benefits? One way is to ask managers and users of the system what they perceive the value to be. For example, a manager can be asked, "If we could give you a new benefit namely, bringing in a service bureau to perform ad hoc querying-what would you be willing to pay per month for that benefit?" or "In your functional area, how much would you pay to outsource a specific task?" By asking people in all functional areas related to the proposed project questions like these, it is possible to estimate the perceived value of possible benefits delivered by the proposed system.

Costs

The costs section should provide the user with a detailed account of the estimated monetary costs of developing the proposed system as well as the internal resources needed to complete and support the project. Costs can be estimated by phase. For example, the estimate for the Analysis phase should be pretty solid; however, estimates for the Design and Build phases will be less precise since there are many more unknowns in those phases at this stage of development. The true costs are never fully known until the system has been in production for one or two years.

“The following should be included in the costs:

- Number of person hours required of internal systems and business people by level
- Cost of external consultants by level, based on hourly or daily rates · Expected number of hours per day or week for each phase of the project
- Expected hardware costs · Cost of software licenses · Networking costs
- Cabling costs
- Number of person hours of internal user resources required

This last item is crucial to the successful completion of any project. The user needs to have a clear

understanding of the need for a person or persons from his or her organization on the development team. These resources should be identified by name: for example, "We need 25 hours with Joe, 10 hours with Susan . . ." and so on. The individuals must be chosen carefully since they will function as the liaison between the analysts and the users. They may be required to devote their full time to the project until it is completed. The liaison person or persons should attend all team meetings and work side by side with the analysts building the requirements documents, designing screen shots, and so on. Their participation in all aspects of system development also helps overcome the us-them attitude of user and analyst and promote a more collaborative effort.

The input from the liaison person will play a major role in setting the direction of the project. When the project is completed, this person will be a more valuable employee to the company since the person will gain an understanding of systems development in general as well as being an expert on the new system. A rule of thumb in selecting this person is that if the users won't miss the person, he or she is probably the wrong person for the job."⁽⁶⁾

Once again, it is important to keep in mind that the analysts and the users are building a contract. There must be a meeting of minds. The users are not just agreeing to pay the analyst but also must be willing to allocate the necessary internal resources to the project. The analysts must be firm about this commitment of resources from the users. For example, if the analysts need 10 to 15 hours of time from several top executives, this time must be set aside. If the resources are not made available or the user attempts to substitute lower-level employees, the analysts must indicate that this places the entire project at risk. These cheaper internal resources then become "pseudo-users," and all the analyst can guarantee is that these pseudo-users will be happy with the results. If the users are not willing to spare the appropriate employees, then the whole system development process is in jeopardy. If the analysts cannot get the right resources, then the analysts need to report that the proposed system may not be deliverable.

IX. Project Organization and Staffing

Identification of the different roles of people working on a project is a frequently overlooked step. There are numerous roles associated with a CADM project:

- Project Leader - This person coordinates all of the other roles and acts as a liaison with the business management in the organization where the project is being done. One of the main responsibilities of the project leader is to make sure that the project plan is being followed. He/she needs to communicate to the project team what steps should be followed and how these steps fit into the overall design plan.
- Systems Architect - This is the technical leader of the project who makes sure that the applications and underlying data are coordinated.
- Business Analyst
 - a) Process Analyst:
This person extracts and analyzes the system requirements with the focus on analysis and representation of business processes.
 - b) Data Analyst: This individual keeps track of and analyzes entity relationships used by the business.
- Logical Data Modeler - This person is responsible for the complete analysis ERD representing all of the business requirements.
- Physical Data Modeler - This individual understands the performance considerations associated with determining the formal data structures.
- Designer expert - This person is responsible for helping implement the project in the software tool.
- Repository Manager - This person takes care of data security and working with different areas of the Oracle Designer repository.
- GUI Design Standards Developer - This person determines the project's GUI design standards by understanding what Designer can easily generate.
- Application Designer - This individual supervises the overall user interface design of the application.
- Application Tuner - This individual ensures that applications are correctly tuned.
- Reports Analyst - This person gathers and documents production and ad hoc reporting requirements.
- Data Tester - This individual ensures that the database meets system requirements.

- Application Tester - This individual ensures that the applications meet system requirements.
 - QA person - This person is responsible for overall quality assurance for the project and ensures that things are done according to the specifications that were laid out.
 - SQL Programmer - This person takes care of complex database and application-level triggers.
 - Application Developer - This individual is familiar with Forms and Reports or whatever application tool is being used.
 - Network Administrator - If necessary, this person takes care of network considerations and the underlying physical network of the project.
 - Systems Administrator - This person supports the operating systems and other systems software.
 - DBA - This person handles backups and recovery and manages database instances.
 - Data Migration Expert - This person manages the process of migrating data from the legacy system to the new database structures.
 - Users - It is important to have users on the development team from each functional area, preferably full time. These individuals should sit in on all meetings throughout the life cycle of the project.
- Many of these roles can and will be performed by the same person. The question arises: How big of a team is desirable? As F. Brooks describes in *The Mythical Man-Month: Essays on Software Engineering* (Reading, MA; Addison-Wesley, 1975), a surgical team model is the best approach. This means that there is one key person supported by as many others as necessary to keep that person productive. On a system development team, a few people do all of the "real" work. Others on the team do support work. We consider it irresponsible and unethical to use high-level development talent for tasks such as typing in entity descriptions or taping together ERDs. The underlying staffing principles here include the following:

- There should be various levels of talent on any team.
- Always use the cheapest, lowest-level talent qualified for any task.
- Never use low-level talent on complex tasks, except for training purposes.
- Use key people to perform key tasks.

On normal, straightforward, low-level development tasks, an expert analyst may be able to complete these tasks in half the time needed by a lower-level analyst. For complex tasks, an expert developer/analyst can often accomplish a task in one tenth of the time needed by a novice developer, assuming that the novice developer could do the task at all. Using the surgical metaphor again, a first-year medical student can suture up a minor cut but no matter how much time the student is given, he or she cannot perform brain surgery. That can only be done by a brain surgeon. Likewise, complex systems tasks require top-level talent.

For the following tasks, it is particularly important to use top-level talent:

- Design of the logical ERD - Lower-level talent can do requirements analysis and the first cut of the logical ERD, but the final version requires much expertise.
- Audit of the logical ERD - This requires a separate highly skilled data modeler other than the ERD designer. It can be a consultant brought in specifically for this purpose.
- Design of the physical database - Physical database design is its own specialty.
- Audit of the physical database - The same level talent applies to this audit as to design of the logical ERD.
- Application design - This person is responsible for the overall user interface for all applications.
- GUI standards development - GUI applications are very different from older, character-based applications.
- DBA - Once the database is set up, this job can be passed to a production/maintenance DBA.
- Internal control system design - This is crucial to the success of the project, particularly on financial systems that need to stand up to rigorous financial audits.
- Change control system - Once the system is in place, an experienced person needs to handle how changes are made and managed.

X. Workplan

The Strategy Document should include a high-level workplan. Other, more detailed workplans will be developed during each phase of the project.

At this point, the workplan should be painted in broad strokes from a top-management perspective. Top management wants to know about the deliverables. The workplan should lay out the tasks to be performed for each major phase of the project. Management will not know when analysis is finished

unless a requirements document is generated. The analyst should make sure to generate deliverables at each project point for management to review.

The Strategy Document should not simply list promises but should tell management what specifically will be delivered, and when, for the entire project. The user should have a clear picture of what the analyst intends to do and how the user will be kept informed of the project's progress. The analyst needs to instill confidence in the user that the proposed process will enable the project to be completed successfully. The user must buy into the process as well as the end result. For example, during the Strategy phase, one deliverable should be the written project plans as they are done. The user should approve each workplan.

The workplan should be an iterative, negotiated document. The first cut should be an appropriate plan based on the strategy already laid out. This plan is a refinement of the contract between the analyst and user. Each should have a clear idea of this agreement.

What should the workplan look like? It should not simply be a Microsoft Project document. A one-page chart should be included as part of the workplan, but this alone is not enough. The full workplan should describe each major process step and the accompanying deliverables in narrative form. This plan will let management know how the analyst intends to accomplish the various phases of the whole project. Management must agree to each part of the workplan.

How should the number of hours be calculated? To do this, the analyst must internally generate another level of detail. Time tends to be notoriously underestimated. Each portion and subportion of the project must be thought through carefully. Although this level of detail is not overly relevant for the Strategy Document, it is useful to have if the user wants to know how the numbers were derived. Common errors to avoid are a tendency to leave out some time-consuming steps and to underestimate how long tasks will take. This more detailed document can be shown to the key user to back up information in the Strategy Document.

"Writing the workplan requires the analysts to make many assumptions. At this stage of the project, they may not know enough to go into great detail. There is still much to find out about the user-site terrain, politics, pitfalls, and so on. It is important to lay out assumptions of the size of the problem and anticipated access to resources. These factors have a direct impact on the workplan portion of the Strategy phase. It is also important to include, as a line item, unexpected disasters. These might include key resource people becoming unavailable for whatever reason and interviews that produce radically differing versions of what needs to be done. Don't underestimate the cost of unexpected disasters. If the development team has already completed a project for this user in the same area, he or she can allocate 20 percent of the overall project cost for unexpected disasters. The project team can perform a risk analysis to estimate the disaster percentage potential. If this is not possible or if this is the first project for a particular user, the development team should plan for 100 percent of the overall project cost for unexpected disasters."(6)

At this stage, the workplan should outline the major phases and deliverables of the project. This can be done with Microsoft Project to produce a one-page document, the goal of which is to educate the user regarding the methods for building the system. The primary goal at this stage continues to be to build the user's confidence in the project team's ability to convert a vision into reality. The workplan should be as detailed as the user wants. The analyst can gauge the level of detail wanted from the user feedback.

XI. Business Impact

How can business impact be determined at the Strategy phase? The analyst will need to hypothesize, "If we had the proposed system in place, what would it do for us?" Obviously, the answer will vary greatly from project to project.

"The bottom line is that the business impact should meet the needs outlined in the motivation section.

The new system should provide a solution to the stated business problems. The analyst should go back to the motivation section and make sure each need is addressed one by one and discuss how the new system will support each need."(6)

XII. Strategy Document Conclusion

The Strategy Document should have a structured conclusion summarizing all of the sections outlined in the preceding paragraphs.

In principle, the Strategy Document should be as detailed as possible. The more analysis that can be included, the better. In reality, however, until the analysis is completed and confirmed with the users, the development team will have an incomplete vision of the scope of the project. In the final analysis, we never know the exact size of a project until it is almost finished. The full scope really can't be understood until the Design phase is complete.

Application Partitioning

It's no secret that smaller applications are much easier to build than larger ones. The number of things to consider increases exponentially as the size of the application increases. A small application with a suite of modules that operates on perhaps ten different database tables can be easily built by a couple of developers in a few weeks. A suite of modules running on 30 to 50 tables will require the efforts of a five- to ten-person development team for several months. A large effort involving several hundred modules operating on a few hundred tables can consume whatever resources are applied to it and may take a year or more, if it is ever completed.

The idea is to break the system into several independent development efforts. There is some cost to partitioning, though. For every portion of a large system that is built, you need to consider how that partition will interface with the other partitions. The system should be partitioned in such a way that the interfaces between partitions are as simple as possible. It is necessary to find a happy medium between many small systems with horrendous interfacing problems and a few large partitions with minor interface problems but that have all the other problems associated with larger systems. In the past, the tendency has been to create very large systems. Our experience has been that projects are usually not sufficiently partitioned.

"How should the partitioning be done? Optimally, partitioning should be done by subject area. The main criteria to use in selecting application partitions is that it should be possible for the partition to be put into production, independent of the other partitions. In most businesses, systems can be partitioned by accounting cycles, such as purchase and acquisition, sales and distribution, or payroll. Partitioning doesn't have to take place only during the Strategy phase. It is possible to partition your system at any point in the CADM process. In fact, the appropriate way of partitioning may not be evident until the end of the Analysis phase. At the end of every phase, the development team should audit the correctness of any system partitioning."(6)

Scope

What we are describing in this book is the process for a complete system design involving a complete replacement of a legacy system or a new system development effort. Frequently, the scope of a project is inconsistent with the full CADM process. For example, in one project that the authors worked on, the mandate was to replace the front end of an existing character-based application. Going back to the users and starting over with a complete Analysis phase was not an option. To further complicate matters, there were other applications outside the scope of this project concurrently using the same data structures. Finally, a cursory review of the database revealed serious design flaws.

Developing the workplan for this project was quite challenging. The Analysis phase was abbreviated and only involved interfacing with existing systems personnel. An audit of the logical ERD and database was performed without a Requirements Document. We had to rely on the in-house developers to act as "human surrogates" for a Requirements Document.

The rest of the workplan followed the model proposed in this book.

The main point is that in many cases the scope of the project may not include full Analysis and Design phases. At the beginning of any project, the analyst needs to evaluate what changes are appropriate to CADM in order to take into account the starting point of the project.

Strategy Document Example

The following example of a Strategy Document uses a small project as an example, because the documentation for a large project can be very long. Nevertheless, this small project illustrates the concepts being discussed.

The example project calls for redesign of a legacy human resources system. This example was chosen because every organization needs to keep track of its employees on some level. The organization in this example is XYZ Company.

The following paragraphs present the Strategy Document for the proposed project.

I. Executive Summary

Using our developing tools, we propose to reengineer the data structure and applications for the XYZ Staff database.

The XYZ Staff database is in need of redesign. The current system is based on a Forms x.x application interface; the database design has some conceptual flaws that require either applications or users to add the same information multiple times into the database. This flawed database structure will increase the overall cost and maintenance of the project because the applications will be harder to write.

We propose to not only convert Forms x.x to y.y but use this opportunity to reengineer the database and clean up its structure.

II. Legacy System Description

The existing system is an version 6.0 database with a Forms x.x front end. The data structure has some conceptual flaws, and the Forms x.x front end will require serious reworking to bring it up to modern standards.

III. Related Projects

This is a stand-alone system that has no need to interface with any other project.

IV. Business and Financial Sponsorship

This is an important project, but unfortunately, its value is not obvious. Funding will come from general department resources. Therefore, the Staff database project will have to be piggybacked onto other projects that have greater political priority.

In this case, we can use this project to set up and design our GUI and our developing standards. Also, we can use the project as a watershed for the application development process using Designer.

V. Motivation

The existing system has numerous conceptual flaws. For example, people are assigned to organizations in three different places in the data structure. Three different intersection tables connect individuals and organizations, so if one individual's relationship spans more than one of these tables, that information must be stored redundantly.

The current character-based application needs to be converted to a GUI environment. Since we are now moving to a GUI Designer environment, we need to set up standards, methodology, CASE templates, and common libraries, and we need to document how applications will be designed in the future.

VI. Project Scope

In this project, we will only make essential modifications to the underlying database structure to minimize the cost of data conversion. We will develop the structure of an Oracle Designer-based methodology and GUI design standards. Then we will build an front end to support required functionality. We will not support any ad hoc reporting or query capability or interfacing with any other XYZ systems.

VII. Solution

We propose to complete this project in two phases. Phase I will determine XYZ's GUI design and application development standards, lay out the application and development methodology, and perform some prototyping on the Staff database. In phase II, we will apply these standards to the problem of reengineering the Staff database.

We will deliver a reengineered Staff database and a set of applications that will act as the front end of that database. Of course, if we change the data structure, we will have to make some modifications to the Reports application that generates the XYZ phone book.

ERD

Figure 3-1 shows the basic data model for the XYZ Staff ID system based on preliminary discussions with management. The diagram indicates that individuals can have multiple relationships of different types within an organization and that individuals can have multiple phones. The diagram also indicates that we will be tracking changes over time (history) for both people and organizations.

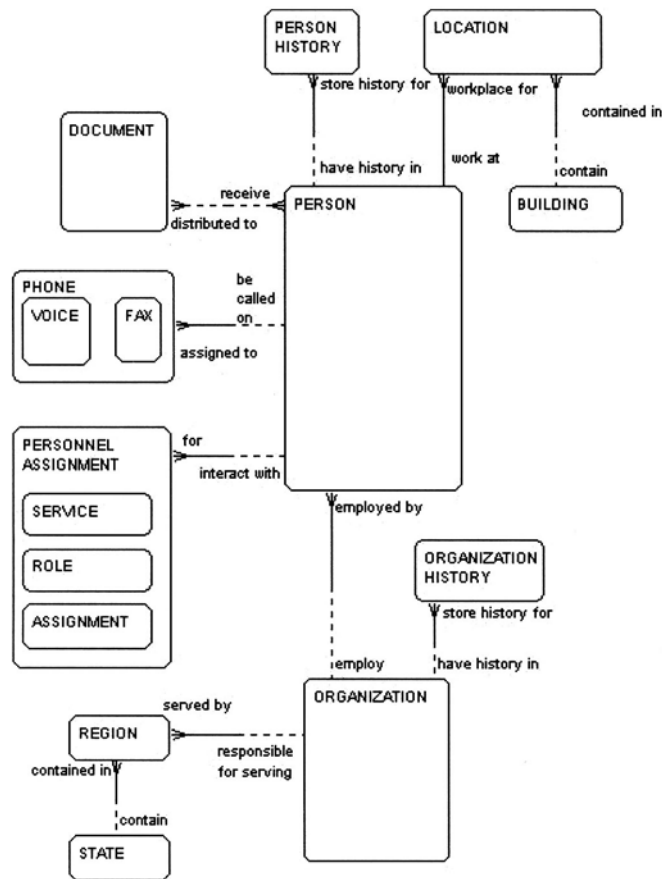


FIGURE 3. Strategy ERD(6)

In the diagram in Figure 3-1, the many-to-many relationship between persons and documents is not resolved, but the many-to-many relationship between persons and organizations is resolved. If this relationship had not been resolved, the diagram would be less clear. Also note that this diagram is limited because a phone cannot be shared by people and any particular phone must be of only one type. However, for a Strategy ERD, this is okay since it is just trying to capture the high-level business requirements: that is, that individuals can have only one phone.

Process Flow Diagram

One of the principal process flows associated with the system is the hiring of employees. This process is represented in Figure 3-2.(6)

modifying these standards to suit your needs.

2. Set CASE Designer preferences (5 to 10 days). There are over 450 preferences to set.

3. Create CASE Designer templates and SQL libraries (5 to 10 days). As we dig this far into the generation process, there will be a clear need to reengineer the templates and libraries released with the product.

Phase II

1. Redesign database (5 days).

2. Create storyboard application (3 to 10 days). We will build the whole application as screen shots so you will have an idea how it will look. If you are not satisfied with our first try, extra time will be required to redo the storyboard.

3. Build detailed design book (2 to 5 days). For each application, we will need to describe how it functions. The level of complexity of the applications will determine how long this phase will take.

4. Module design within CASE Designer (10 days). We are guessing that we will have 50 modules to build. This will take some time.

5. Generate and modify modules (10 days).

6. Engineer changes into templates, libraries, or module definitions (5 to 10 days). The time needed for this step will depend on the number of modifications required for each module. Only complex modules will need significant modification.

7. Build system documentation (5 days). Most system documentation will already be included in the design book or can be generated using CASE Designer. We are allocating a week for cleanup and quality assurance (QA).

8. Build user documentation (5 to 10 days). With user feedback, this step always takes longer than originally planned.

9. Testing and QA (5 to 10 days). We will audit the applications to ensure that they meet the design specifications of the design book.

XI. Business Impact

Because the scope of this project is so small, the business impact was discussed under Benefits (see Section VIII).

XII. Strategy Document Conclusion

By setting up GUI and design standards that can be used for future system development and restructuring the existing database, we will make the XYZ Staff system more efficient and user friendly.

Modifications for Smaller Systems

"A full-blown Strategy Document is appropriate only in very large projects costing a million dollars or more. However, the Strategy phase can certainly be pared down to suit smaller projects. You can stratify the size of the projects being discussed, as follows:

- Type 1 Multimillion-dollar projects requiring person years of effort taking place over months or years
- Type 2 Medium-size projects, including stand-alone systems such as payroll and purchasing systems, and systems requiring integration with existing systems
- Type 3 Small projects that are short term and require limited budget and resources

(Note that any project requiring integration with an existing system can be viewed as one size larger. In other words, a small project requiring integration with a functioning system should be considered a medium-sized project, and so on.) For very small projects, the Strategy and Analysis phases are combined. For example, when building a help desk system, all that is necessary is a few-page write-up indicating the project scope and cost estimate, a Strategy ERD, and perhaps a process flow diagram. The result will be a Strategy Document with some analysis and design specifications all rolled into one. From this, the project can move to a rapid prototyping environment. With the sophistication of existing tools such as Oracle Designer, the application can be built so quickly that if it is not satisfactory, it can be rebuilt and quickly moved through the refinement process. Descriptions of the legacy system and political environment also are not necessary for small projects. For medium-sized systems, a Strategy Document still is necessary. However, not as much time needs to be spent on this since there will be fewer users to interview. Also, since the project is smaller in scope with less money involved, you will not need to justify the plans as carefully at this stage as would be necessary in a large project. For small and medium projects, an executive summary is needed only if the Strategy Document is more than ten pages long. Legacy system and political environment descriptions may or may not be needed, depending on the specific situation. Within this book, most of the information pertains to projects of type 1 in the preceding list. However, modifications of the proposed methodology for small- and medium-sized projects will be mentioned at the end of each phase, as in this section."⁽⁶⁾

When Is the Strategy Phase Complete?

How do you determine when the Strategy phase is finished? A rule of thumb is that it is finished when all major players believe it is finished. This consensus is formalized when everyone involved signs the Strategy Document. There are no objective criteria for determining completion. However, it is important to gather and verify as much information from both users and in-house systems staff as possible. If the analyst does not get an approval from the primary user, then the definition of whether or not the system is complete is ambiguous. The analyst must put in writing that if access to the proper individuals is denied, he or she cannot be held responsible for the outcome of the total project. It is up to the analysts to write a complete Strategy Document that covers all sections of the project. This should be a joint effort with the user community. This contract is executed when all parties sign, thus binding both analysts and users to the stated scope and deliverables.

Chapter 3 : Analysis phase

Gather the requirements while you may. The old system is still a flying. And the same project that's on time today, tomorrow will be dying. (Apologies to Robert Herrick.)

Analysis is the process of gathering and analyzing system requirements. At this point in the system development process, the analyst should have the following idea uppermost in mind: "It is not possible to meet a user's need that was never discovered."

"Consider this real-world example of lost requirements. The Analysis phase was almost finished when the analysts were approached by one of the client's legacy system programmers and told the following: "There are elements written into the programs that allow us to do some types of business in New Jersey and not in Pennsylvania. Most of the users are unaware of this. Where are these requirements being written down?" In this case, the confusion occurred because the project leader had not thought through the Analysis phase completely prior to its execution."(6)

It is very easy to just begin talking to people and gathering data. However, the analyst must constantly keep in mind the following questions:

- How will the system designers find out everything that they need to know to complete the project?
- Are the resources adequate?
- Will the analysis process truly capture all system requirements?

The goal at this point is straightforward: namely, to execute the analysis plan. However, throughout the Analysis phase, the analysts need to think very carefully about the process that was outlined in Pre-Analysis.

Analysis consists of two parts: information gathering and requirements analysis. Information-gathering activities, such as conducting interviews, implementing questionnaires, holding JAD sessions, and reviewing the legacy system, are performed first. Analysis begins as soon as some information is gathered; you do not wait until all of the information is collected to begin analysis. As you are gathering information, it is constantly being analyzed. Once all of the information has been gathered, a final assembly of the information analysis must be done. Towards the end of the Analysis phase, inconsistencies among requirements are identified and resolved and a coherent statement of the new system requirements is formulated.

Deliverables

The ultimate deliverables for the Analysis phase will be discussed later. Independent of this final Analysis Document, specific interim deliverables are produced during the Analysis phase. These interim deliverables act as supporting documentation for the Analysis Document but may not appear in its final version. The deliverables for the information-gathering process are the following:

- Interview notes · Narrative text
- Lists of documents reviewed
- Worksheets documenting information-gathering activities so that, at a future point in time, a review of what work was accomplished can be compiled

It is important to carefully document information-gathering activities. Of particular importance is keeping track of interview notes and documents reviewed during system analysis. The goals of information-gathering activities are not just to gather all of the relevant information but also to adequately organize and store this information for later review and retrieval.

Overview of Information Gathering

Analysts traditionally follow a two-step process:

1. They talk to users in order to understand what they want and need.
2. They build ERDs, function hierarchies, and prototypes.

Steps 1 and 2 must be linked together, however. Creating a transition step-the delivery of the Requirements Document-provides a means to cross-check the functional models and ERDs, and not just directly with the users but with a document the users approved weeks ago, about which they said, "Yes, that is the way we want it done."

This extra step also gives the users something to compare with the system function hierarchy and ERD. They first compare what they want done with their requirements list. Then they compare this requirements list (which they have already approved) to what is in the ERD, function hierarchy, and any prototypes that have been made.

During information gathering, the analysis team should be focused on determining the system requirements. If the information gathered is incomplete or inaccurate or does not reflect the users' needs, the system to be implemented will be a failure. It is important to spend an adequate amount of time to gather this information completely and effectively. In general, information-gathering tasks require both a strict information gathering process and a separate process to analyze and document the information collected at each information-gathering activity.

Devices such as prototypes, screen shots, preliminary database models, and storyboarded applications are appropriate only insofar as they serve as vehicles to communicate back to the users what the analysts understand their needs to be.

This feedback to the users should be done at the "unit" level, interview by interview and user by user. For example, in an individual user interview, the analyst tries to collect as much information as possible. Then the analyst examines the information gathered and generates informal ERDs, process flows, or other appropriate representations to effectively feed back to the user, in an organized way, what was learned in the interview. Ideally, this unit analysis activity takes place with the help and hands-on support of the user. Unfortunately, sometimes users have neither the time nor the inclination to be this involved in the system development process.

After the unit analysis is performed, the information is passed back to the user for approval of both the interview notes and whatever analysis of the interview was performed. Until the user signs off, the interview is not complete.

The analysis process can then proceed. The analysis plan prescribes various information-gathering activities that need to be performed in a particular sequence and the degree of parallelism of these activities. Each information-gathering task follows the same process. Information gathering and unit analysis are described in the remainder of this chapter.

Eliciting Requirements from the Users

This section discusses the various methods of gathering the information necessary to understand the business area and identify requirements. These methods include:

- Interviews
- Questionnaires
- Electronic communications
- Joint application development (JAD) sessions

Interviews

Interviews that are conducted carefully can go a long way toward eliciting the necessary system requirements. The standard procedure for gathering user requirements is to interview the business users. This may sound like a straightforward and simple task, but the methods used to obtain and organize the information can greatly affect the results.

Open-Ended Interviews

"Open-ended interviews consist of one or more analysts and interviewers sitting down with a user and asking questions. It is critical to constantly keep in mind the nature of the analyst-user interaction and the purpose of the interview: namely, to understand the business and identify requirements. This can be supported in several ways: through good questions and comments, productive feedback, and a preexisting structure for the information to be gathered.

When conducting interviews, it is very useful to bring a person other than the interviewer to act as a scribe for the interview. In this way, the interviewer can concentrate on the questions being asked and on providing appropriate feedback, as discussed later in this section. A logical choice for this role is a junior analyst. He or she will gain experience in how to conduct an interview. The senior analyst can then review the write-up of the interview shown to the user for final sign-off.

QUESTION SYNTAX IS IRRELEVANT Much has been written in various fields about the syntax of the questions asked in an interview and how this affects the information gathered. These studies typically stipulate that questions must be open-ended so as not to bias the feedback, and they conclude that careful attention must be paid to the exact words and syntax of the questions in order to get the most accurate answers. These conclusions were reached by looking at standardized exams, police interviews, psychiatric interviews, and teacher-student interactions.

However, there is a substantive difference between the interactions in the standard research settings and those in a systems development environment. The goal in system development is to give and receive information. The relationship between interviewer and user is cooperative, not adversarial, and there is no attempt to coerce or trick the person being interviewed. Both the analyst and the person being interviewed are working toward a common goal. For example, it didn't matter which of the following two questions the interviewer asked:

"Do you want a printer?" (yes/no question)

"How would you like to handle your printing needs?" (openended question)

If the syntax didn't make a difference, what did matter in conducting a successful interview? The conclusion reached was that certain interviewer behaviors and strategies were the key factors.

LISTEN, THEN TALK. The most important factor observed from the hours of interview tapes was that the interviewers who were most successful placed themselves in a mode to receive information and stayed locked into this mode. Their primary goal was to listen, not give information. To do this, they displayed several key behaviors, some of which are covered in the literature on active listening:

- They asked good questions.
- They gave encouraging nonverbal feedback (such as note taking, head nodding, and "uh-uhs").
- They let the user say what he or she wanted to say without interruption until finished.

Interview questions should be designed to open up topics for the user to talk about. The comment "Let's talk about printing" could elicit the same feedback as yes/no or complex questions about printing. The important point is for the interviewer to always keep in mind the specific setting and situation of the interview in the business environment. The nature of this interaction is that information can go only one way at a time. Someone must be giving information, and someone must be receiving information. If both the analyst and user are trying to give information at the same time, then no one is listening.

USE FEEDBACK. Good listening behavior, however, is not enough. Attentive listening and "uh-huhs" do not communicate to unhappy users that they have been heard and understood. The most successful interviewers repeated back what they had heard, using statements like "Let me see if I understand what you are saying" or "From what I've heard, this is what you are looking for." These interviewers were able to get much more information from the users. There was a direct correlation between using these feedback strategies and getting more information. Finally, a key question that must always be asked is "Is there anything you would like to add?" This makes the person being interviewed feel that the interviewer has heard and thought about what has been said and forces acknowledgment that the user has no more to say.

USE POWER CAREFULLY. Another important issue in conducting successful interviews is that of power. In a typical interview situation, there is a clear delineation of who is in the position of power. This is not the case in an analyst-user relationship. The optimal arrangement is a relative parity in power positions between the interviewer and the user. In general, the user needs to feel comfortable enough to talk about what needs to be discussed. If a user is insecure, it is the job of the interviewer to put the user at ease. Ways that this can be accomplished include conducting the interview on the user's turf (for instance, in the user's office) and using nonverbal cues. Nonverbal cues may include note taking, head nodding, body position, and voice volume. These cues can be used to manipulate the power structure of the situation to suit the needs of the interviewer.

For example, if the user isn't letting the interviewer get a word in edgewise, the interviewer can take control by standing and walking to the white board or using paper and drawing or reviewing what has been discussed. This puts the interviewer in the place of highest power, since all attention will be focused on the person at the board. At that point, the interviewer owns the situation and is in control. Less severe power-gaining measures include controlling the actual volume (voice level) of talk. Also, the interviewer can remain in an information-receiving mode but encourage an alternation of discussion between himself or herself and the user. If done correctly, this can make for a very effective interview. Body position can also influence the power balance. If the interviewer is sitting in a relaxed pose, this is a higher power stance and tends to put the user in a weaker position. In the opposite situation, where the user is reluctant to talk, the interviewer needs to give the user more power. This can be done with body language: for example, the interviewer can sit up in an attentive pose, waiting to receive information. Gentle cues ("Tell me about. . .") and questions can draw out the reticent user and extract the necessary information.

A related issue to that of power in analyst-user interviews is that of gender. Because of deeply rooted social norms and perceptions within society, the gender of the interviewer and user may play a role in the success of the interaction. The interviewer simply needs to be aware of this and make slight adjustments in the power structure of the interaction. If a male interviewer is dealing with a female user accustomed to a male boss, he may want to speak quietly and take lots of notes to facilitate better transfer of information. Conversely, a female interviewer with a male user accustomed to being in control may want to use more power-increasing strategies in her interviewing.

PROTECT AGAINST BIAS. One factor to be very careful about in interviewing but which can be quite difficult to detect is interviewer bias. The potential for bias on the part of the interviewer in an open-ended interview is even greater. The only effective way to guard against bias is to use several different analysts who will hopefully have different biases. By analyzing all of their information, a balanced picture of user requirements can be obtained.

Another way to help prevent interviewer bias is to send two analysts to conduct an interview with one user. In this way, their individual biases will have less effect. An added benefit of this strategy is that one analyst can be assigned to interview the user from a data-centric perspective and the other from a more process-centered perspective.

STRUCTURE YOUR INFORMATION RETRIEVAL. So that as many requirements as possible are uncovered and the validity of the information is explored as thoroughly as possible, three topics must be addressed during user interviews:

- What each user does
- How each user does what he or she does
- What each user needs to do his or her job

It is also important to discuss these topics, which don't relate directly to the user's current system:

- Improvements that the new system could supply
- Topics outside the range of the user's immediate job function

When asking what users do, the analyst should have them list their tasks. For each task noted, the analyst then should determine the process flow. The process flow for a task is simply how people do their jobs. Each task is further refined into subtasks. In most cases, a subtasks(functions) become a screen module if the user needs to create update, or delete information. A function becomes a report module if the user needs to retrieve the information on the screen or on paper. If the user performs quality checks on the data, the function might become an integrity constraint.

The third topic, what users need to do their jobs, includes determining what functions and data users need. The decisions a user makes during the day are often critical issues when determining the requirements of the system.

After the first three topics have been covered with the user, what the user needs in the new system can be addressed. Now that how the existing system works has been detailed, areas for improvement can be discussed.

Finally, the analyst should ask the user for comments that do not relate directly to the user's tasks and job. These not only shed light on processes that might affect the user indirectly, but also may open up ideas for future work. For example, a person might explain that checks are never issued on time. This may be out of scope for the current project, but an attempt to work on this related project could be made in the future.

Following this interview procedure will allow you to question users in a way that stimulates discussion. Users discuss what they do every day with coworkers and friends. In most cases, you will not get a constructive response if you ask a person, "What are the requirements of the new system from your point of view?" It is much better to put the system in perspective and have the users tell you what they do, how they do it, and what they need to do their jobs.

GROUP QUESTIONS INTO CATEGORIES. A skilled interviewer may be able to elicit a great deal of information, but without some type of format or structure, the value of the information is greatly diminished. A skilled analyst organizes his or her questions into related topics.

These structurally related questions typically start with a structural identifier. The interviewer introduces topic A and then provides contextual structural cues relating to the question to reduce the probability that the question will be misunderstood. In this fashion, subtopics A1, A2, and so on can be covered in depth. The interviewer builds a tree of information with three types of questions:

- Validation questions For instance, "Is this what you are saying? Have I got this right?"
- Horizontal questions For instance, "You've told me about A1 and A2. Is there an A3? Are there other areas about A that we need to discuss?"
- Vertical questions For instance, "Tell me about A1. Is A1 important? How should we handle things associated with A1?"

The skilled analyst is able to organize information as it is being gathered into a structural hierarchy. The analyst begins with a list of questions and moves both horizontally and vertically through the topics. The responses can be organized into a tree structure where one topic (such as A) is the parent, and each parent topic has children (A1, A2, A3). These children are siblings and follow a horizontal format. Each child can then be further broken down into more layers of subtopics in a vertical format (A1 .1, A1 .2, and so on). Each horizontal and vertical topic is explored until a termination point is reached on every set of siblings in the tree. This point is reached when the answer to the question "Is there anything you would like to add?" is "no." In writing up this information, horizontal and vertical lines can be drawn at the appropriate termination points to indicate that the topic has been exhausted.

WORK FROM AN INFORMATION TEMPLATE. Having an internal information template has been found to be one of the key factors in distinguishing an experienced systems person from an inexperienced one. A large body of research on schema theory suggests that learning and comprehension of new information is strongly affected by whatever preexisting related information the learner brings to the task. This can work both ways in an analyst-user interview. The experienced analyst brings a whole range of information on system development and business applications to the interview. At his or her disposal will be many detailed questions about various types of systems that will help shape the interview and get at the desired information. Users, for their part, bring knowledge of their business and what they want the new system to be able to do. Also, they may have some expectations regarding the questions the interviewer will ask. If there is a conflict between the existing schemas of the analyst and the user, the appropriate information may be difficult to obtain.

At the highest subject level, the analyst's internal template, or schema, may be the following question: "If I'm going to build a system, what are the hardware and software requirements?" For an inexperienced analyst, this might be the whole template. A more experienced analyst would have a much more extensive, complete, and detailed set of questions to ask.

For example, to get at the system requirements for performance, a good analyst would know to ask the following questions:

- What is the acceptable time lag for each transaction?
- How up to date must information be in each context? For example, for trend analysis, work on a data warehouse that is updated monthly might be adequate. For a customer billing inquiry, information may have to be up to the second.
- What is the cost of the system going down for 1 minute, 10 minutes, 1 hour, 4 hours, 1 day, etc.?
- How many transactions per hour will the system need to support? · How many users will be on the system at one time?

The more complex the structural template the analyst brings to the interview, the more likely he or she is to get all the necessary information to eventually create a system that successfully meets the users' needs. The

ultimate goal is to get the maximum amount of information on the user's business, existing system, desired changes, and hoped-for aspects and features of the new system.

In reviewing the research videotapes in the author's doctoral study, three levels of analysts emerged based upon their interviewing procedures:

- Inexperienced: These interviewers took notes and did a lot of head nodding and saying "uh-huh" but elicited only a fraction of the potential information from the client.

- Intermediate: These interviewers followed up on particular threads of information by asking some questions but let the conversation wander. If the interviewer was lucky, the appropriate information was elicited, but it was disorganized, which made it easy to miss important points.
- Experienced: These analysts walked into interviews with an information template in their heads that allowed them to work through a systematic hierarchical structure of information retrieval. The user can provide a great deal of information, but the information given will not be generated in a structured format unless the interviewer provides this structure. For example, the skilled interviewer will already have identified three important areas (A, B, and C) that need to be discussed and will begin by saying, "Tell me about A." From that prompt, topics A1, A2, and A3 will emerge. For each area, the experienced interviewer will lead the user through the information in a structured fashion, and at some appropriate point, the interviewer will review each point with the user to ensure that all the relevant information is elicited. The interview procedure allows the analyst to develop a list of requirements and show each user what he or she said. It gives the analyst a document that can be given back to each user. The analyst can say, "This is what you told me," and ask, "Do I have it right?" Recording the details is important in big projects, where many ideas that may seem little at the time may fall through the cracks. The Requirements Document is used to explain to the user, "You want the existing system with these changes. We understand the existing system does these things." It does not get any simpler; the analyst has just outlined the scope of the project with the user's own words.

TECHNIQUES OF THE GOOD INTERVIEWER. To summarize, in order to be a successful interviewer, you should use these four important techniques:

- Go into the interview with an internal template of topics and questions relevant to the specific business situation.
- Ask the kinds of questions designed to elicit the maximum amount of information in a structured format.
- Listen actively and effectively, providing useful verbal and nonverbal feedback.
- Provide a hierarchical structure within which the information can be elicited and organized.

When all these elements are in place, the information you gather from the interviews will provide a solid base from which to start system development and will go a long way toward ensuring that users are entirely satisfied with the finished product. In addition, carefully documented and reported user interviews can be invaluable in determining the source of potential problems with the finished product and ways to solve them."⁽⁶⁾

Questionnaires

Unlike open-ended interviews, the question syntax in questionnaires greatly influences their effectiveness. In questionnaires, significantly different results can be obtained depending upon the wording of the questions. The order of both questions and response selections can also affect responses. Items listed first tend to be selected more often. For instance, if asked "Which online service would you rather have access to: CompuServe, Prodigy, or America Online?" more users would choose CompuServe simply because it is listed first.

If questionnaires are administered face-to-face, the issues of bias mentioned for interviews should be taken into consideration.

A potential problem with questionnaires is not asking the right questions. Ask any teacher or professor how difficult it is to write tests and exams.

Think back to the last few times you received a questionnaire in the mail where the questions did not make sense. When composing questions, you must consider the complete range of people who will be answering them. Questions appropriate for managers may not be appropriate for end users. You may need to develop different questionnaires for different classes of system users.

In developing questionnaires, you should subject the questions to extensive testing. Present sample questions to a few users face-to-face to ensure that users interpret the questions the way you intended.

Once questionnaires have been designed and distributed, the results need to be analyzed. The analysis of the answers is a complex task that requires a skilled statistician.

Electronic Communications

Electronic information can facilitate various portions of the analysis process as well as other phases within CADM. Two are presented here.

E-mail list servers and groupware products can be used for discussion among users about the system being designed. For example, a web-based application can be set up to debate the features that the new system toolbar should include and whether the organization should go to the expense of including a user-customizable toolbar. Such an application, where any user could post or query messages, would be simple to set up. All messages would be visible to everyone using the application. The second way electronic information can be used to assist in the gathering of user requirements is by giving users direct access to the list of requirements being compiled. Users can review and respond to specific items on the list. A web-enabled application can be set up to access the Repository. This would allow users to see the listed functions and their associated requirements and enable them to suggest new requirements or comment on existing requirements.

Joint Application Development (JAD) Sessions

"A JAD session utilizes a workshop setting where business and technology professionals participate in the planning, analysis, and design of a system development effort. JAD sessions produce the best results when the number of participants does not exceed 20. JAD sessions should be attended by representatives from all interested groups. The organizers should ensure that the group is reasonably balanced in terms of user level.

In order to be effective, JAD sessions must have some specific purposes and goals. They should not be used for general application development work. They are very expensive and frequently not efficient.

A JAD session is an inherently political event that may often involve more political posturing than problem solving. Avoid holding a JAD session to discuss a topic when there is no decision to be made. In such a case, the main function of the session will simply be to allow users to feel as though they have been heard. Often, the outcome of these unfocused JAD sessions is the appointment of a steering committee to perform the work that the JAD session was supposed to accomplish.

If you are going to hold a JAD session, it must have a precise agenda and goal. The more precise the focus, the more successful the JAD session is likely to be. To keep JAD sessions from wasting time, a useful strategy is to schedule them before lunch. Participants are sharper and more focused in the morning, and after one or one and a half hours, they will be ready to leave. In the authors' experience, rarely has anything useful ever happened in a JAD session after the first one and a half hours.

A successful JAD session must have a trained and experienced facilitator whose job it is to work with the project team in developing the purpose, scope, and deliverable in advance of the session. The facilitator must manage the group dynamics, enforce the ground rules, and adhere to the agenda. The purpose, scope, anticipated deliverables, and the agenda should be documented and distributed to all participants well before the scheduled meeting time. The approximate amount of time allocated to each agenda topic should be indicated.

JAD sessions are useful when conflicting user requirements are encountered. A JAD session is the best place to resolve such conflicts. If there are very strong concerns, the proposed session can be used to communicate these concerns to all groups.

An additional function of a JAD session can be to convince obstinate individual users who believe they speak for a large number of other users that they may be mistaken. The JAD session can be a vehicle for overcoming the objections of one or more individuals by demonstrating a general consensus on a particular issue."⁽⁶⁾

Modifications for Smaller Systems

You don't need to make any modifications whatsoever for smaller systems for this part of the analysis process. Information gathering is essential to any project. All that is different for a smaller project is the amount of information gathered. Procedures, interview notes, user feedback, sign-off, and so on should all be conducted in the same manner no matter how small the system.

When Is Information Gathering Complete?

Information gathering is an ongoing process throughout the System Development Life Cycle. However, it is easy to fall into the trap of "analysis paralysis," which can stall a project indefinitely. Often, what harms a project at this point is not the information that has been gathered but information that has not been kept track of properly after it has been gathered. Nothing is more infuriating to users than having a second or even a third group of analysts ask them the same questions. If a user ever asks, "Don't you people ever talk to each other?" you can conclude that the information-gathering process is flawed.

Being meticulous in all aspects of the information-gathering process is an important attribute. Obtaining requirements from so many different sources will, of course, result in some duplication of effort. However, the cost of missing an essential requirement is much higher than the cost of collecting a few requirements more than once.

This is a test of the Application Development System. Had this been an actual application, your data would already be corrupted and your organization would be plunged into chaos. This is only a test this point, the system has been built, unit tested, and passed on from the developers to the test team. If you have followed the CADM process carefully, the Test phase should run smoothly.

Some of the activities and quality control executed during earlier phases that contribute significantly to the success of the Test phase include the following:

- Users approved the Requirements Document at the end of the Analysis phase.
- After the logical ERD was created in the requirements analysis portion of the Analysis phase, you worked with one or more senior modelers to carefully audit the logical model.
- The system storyboard went through a formal acceptance testing process with the users.
- Requirements were mapped to functions and modules.
- Column-level usages were specified for each module and reviewed by senior team members.
- The design book was audited against the system requirements.

One of the most important features of CADM is that as you move into the Test phase you have a complete audit trail that allows you to logically follow a system requirement gathered in the Analysis phase all the way through to the completed system. Because of the way the system was built, you know that it meets the stated system requirements as you understand them. Every step in the process has been checked and audited throughout the system life cycle. Theoretically, all that is left in the final Test phase is to conduct integration, system-level performance tests and user acceptance testing for the application.

However, as you have proceeded through the CADM process, everything undoubtedly has not gone completely smoothly. User requirements may have changed over time. Users may have changed. The scope may have changed. The business may have changed. In the course of the project, your thinking about how the requirements should be distilled into an Analysis Document may have changed. The design evolved as you moved through the Design and Build phases. For these reasons, many of your earlier audits and deliverables may not still be valid. In the development of the test plan, you need to carefully consider each of the major deliverables and decide what level of testing is appropriate.

Various system tests must be performed. Unit-level testing was already performed in the Build phase. However, you need to ensure that the new system interacts smoothly with existing systems, handles business transactions adequately, and performs adequately with the organization's full production load. The goal of user acceptance testing is to help uncover problems in the user interface and inadequacies in the GUI design. User acceptance testing should not be performed until after the system testing is mostly complete and the development team is confident that the system is basically sound.

Chapter 4: Overview of the Test Phase and Implementation Phase

Testing Phase

The Test phase is the point at which the new system is formally tested. In the Test phase, you develop a test plan that should describe not only tests to be run but also how test failures or variances will be handled. Especially for a large system, it is not practicable to wait until the end of the Build phase to start developing this test plan-there must be some overlap. As each module is tuned and passes unit-level testing, it is stable enough so that formal tests can be planned.

It is not necessarily true that every test failure or variance leads to modifications to the system prior to production. Within the Test phase, it will be necessary to reaudit the design process, perform system- and user-level tests, audit the quality of the documentation, test the internal controls and backup and recovery procedures, and in all ways ascertain the fitness of the system to move into a production environment.

Test Plan

The lead QA person needs to develop a test plan. Users need to be involved to identify the test cases, and they can write the test scripts and expected outcomes. There are two components to a test plan: the approach and the design. The approach includes the testing techniques, testing tools, roles and responsibilities, method of error detection and logging, change control process, retesting process, and testing environment. The design part of the test plan includes the test objectives, cases, and scripts. The test plan is the framework in which the different levels or types of testing are performed, i.e. system testing and user acceptance testing. You should develop a test plan for each type of testing. The test plan must encompass the following elements.

- System tests:
 - Audit of major CADM deliverables Make sure early phases were completed correctly.
 - Integration tests. Make sure applications work together smoothly.
 - Transaction flow tests Make sure business transactions meet business requirements.
 - Stress tests. Make sure the system can handle a realistic production environment.
 - Data migration validation Make sure the legacy system data is moved accurately into the new system.
 - Backup and recovery tests. Make sure the system is adequately protected against potentially catastrophic system events.
 - Internal control evaluation Make sure the new system is secure.
- User acceptance tests:
 - Small pilot lab tests Have a few users try out the applications in a controlled environment.
 - Training sessions Conduct training sessions that also serve as user feedback sessions.
- Training and documentation:
 - User training finalize the training material and train the trainer(s).
 - User documentation. Ensure that user documentation is accurate.
 - System documentation. Finalize the system operations manual and the disaster recovery manual.
 - Help desk. Make sure the help desk personnel have the documentation and training necessary to support customer problems.

System Testing

System testing must validate two aspects of the system: the database and the applications. However, there is not always a clear distinction between the two. For example, are database triggers part of the database or the application? For the purposes of this discussion, the database will be defined to include all tables, table structures, and database constraints. Since the Analysis phase included a logical audit of the ERD and the Design phase included an audit of the physical database design, you do not need to test the structure of the database at this point. The physical structure has already been checked. However, if modifications are made, these will have to be tested.

With regard to the application, several testing issues must be addressed. The integration of the applications must be clean and seamless. The most complex and failure-prone portions of the application are the interfaces between parts of the new application and the existing systems. Even though all systems have passed unit testing, you need to ensure that all of the interfaces are correct.

The essence of system testing is not to test the individual modules; that has already been done. Instead, entire business transactions must be processed through the system.

You can probably assume that not all of the business rules have been implemented as triggers and constraints. Not all of them could be, or the system would not run, would take years to code, and would never deliver adequate performance even if it were coded. It is therefore necessary to write some procedures that validate the business rules not implemented as constraints. For example, in an insurance company, a policyholder is not allowed to have two policies in force for the same coverage at the same time. A business rule set up as a trigger to test for this condition might make any modifications to the insurance policy coverage table unacceptably slow.

The application must also be tested at full production loads-not just at today's production capacity but also at projected levels of capacity for the life of the system.

There are many ways of checking the application portion of the system (that is, the code). The principle behind good testing is the one auditors use to find errors in large accounting systems. Many tests are run looking for the same errors in different ways. The logic is that if errors are not caught one way, they will be caught in another. Therefore, applications can be tested by running lots of little tests and looking for evidence that shows how well the system is working.

Auditing Major CADM Deliverables

During each stage of the CADM process, some tests will likely fail, new requirements will crop up, there will be scope creep, realignment of priorities, and new areas may need to be added to the system being built. Managing and controlling these changes in a thoughtful and systematic way can mean the difference between system success and failure.

At this point in the system development process, it is important to take another look at all of the major deliverables in the CADM process to decide what auditing is necessary.

STRATEGY DOCUMENT. If the system being delivered is significantly different from the proposed system, that difference needs to be documented. All that can really be done is to identify where objectives have not been met or where other valid objectives have been added. You may find that some portion of the system was overlooked and more work needs to be done. At this late point, anything missed will probably not be incorporated until the next version of the product.

REQUIREMENTS DOCUMENT. A full audit of the Requirements Document was performed at the end of the Analysis phase. However, as mentioned earlier, the Requirements Document has probably changed significantly since the Analysis phase. There are two alternatives:

- Audit the process used to allow changes to the Requirements Document.
- Reaudit the Requirements Document in the same way as at the end of the Analysis phase.

If you were careful in your original audit of the Requirements Document and have controlled and documented all changes to the Requirements Document, you can be reasonably confident that your Requirements Document reflects your best current understanding of the system. The only part that would remain to be checked is whether or not the user requirements themselves have changed during the System Development Life Cycle. If they have, you may need to modify major portions of the system to make the system useful. One of the most difficult decisions to make is whether to go back and make substantive changes to the system before delivering version 1; however, it is better to delay a system than to deliver a grossly inadequate system. Of course, such an action should be undertaken only if absolutely necessary. In general, it is better to deliver something than nothing.

If your audit of the procedure used to change the Requirements Document indicates that changes were not carefully controlled, it may be necessary to redo significant portions, if not all, of the Requirements Document audit. However, if such an action is necessary, the entire system is at risk. This is tantamount to acknowledging a lack of confidence in the foundation of the system. If the Requirements Document audit fails, everything done after Analysis may need to be redone.

DESIGN BOOK. Of course, the design book will change a lot over the project life cycle. It has had periodic reviews and updates and remains a working document from the Pre-Design phase through the Build phase. The most important point of your audit is to make sure that the design book is up to date. You cannot test the new system without an accurate, up-to-date design book.

THE SYSTEM. The final deliverable is the system itself. Testing the system is what people traditionally think of as the work of the Test phase. As when testing other aspects of the CADM process, the key to good testing of the system is to perform multiple tests using different individuals and approaches to try to determine that the developers were careful and conscientious in conforming to standards and satisfying the system requirements.

Performing Integration Tests

As business transactions move through modules, they usually need to interact with other modules. For example, in a PO (Purchase Order) system, when a PO is initiated, it should show up in the approver's module for approval. To test this, you can enter a PO into the system to verify that it shows up in the approver's module.

You need to identify every such interface point in the system. In general, such interface points are easily identified through physical process flows. Modules naturally flow from one to the next.

The testing of interface points involves the entry of information into one module and the retrieval or manipulation of that information by another module. The test should involve not only observation of the data through the modules, but also direct observation of the underlying database tables using SQL.

Performing Transaction Flow Tests

Transaction flow tests are an extension of integration tests. In this process, you walk entire business transactions through the whole new system. For example, in a PO system, you can follow the progress of a specific PO filled with sample data from its initiation through the approval process and the receipt and distribution of the goods. Transaction tests can be performed using only the interface with the modules, assuming that module-level tests and interface tests have been performed. The transactions must simply be walked through the entire process.

To adequately test the system, a small amount of sample data also needs to be used to test the entire process. This allows you to test not only the flow of transactions through the process flows, but also the interaction of those process flows with each other.

Performing Stress Tests

Just because a system works on sample data with one user on a dedicated machine does not mean that the same system will perform adequately in a true production environment. The system needs to be tested using realistic production loads to simulate the actual number of users in their actual locations, with production-level database sizes and realistic transaction rates (transactions per second). The stress testing should be done not only on current production loads but also on projected future loads.

Another type of stress condition that must be tested is multiuser locking. Sometimes locking problems are not discovered until several users are simultaneously trying to interact with the system. You need to either simulate or test a multiuser environment. However, it is likely that until the system is in the early stages of Implementation, it will be difficult to do this.

Validating Data Migration

Making sure that data migration is accurate is a very complex step that should be performed table by table, subject area by subject area as data migration takes place.

Performing Backup and Recovery Tests

Backup and recovery procedures are crucial parts of any organization's system, and the testing of the backup and recovery system is just as crucial. If the system is supposed to support operation 24 hours a day, seven days a week, with no more than 20 minutes of consecutive down time, then a tester should be able to walk into the computer room and pull the plug, and the personnel on-site should be able to bring up the system again within the allotted time.

Realistic backup and recovery tests should be conducted without allowing the operations personnel to prepare for the tests. Real disasters rarely occur with prior warning. You can't realistically test a backup and recovery system by scheduling the test at a convenient time for systems people two weeks in advance. Each exposure that the backup and recovery system is supposed to protect against should be individually tested.

Evaluating Internal Controls

Testing the internal control system is much like testing the backup and recovery system except that there are many more exposures, most of which are, in general, less severe. The principle is the same: you should simulate each exposure, from incorrect data entry to underlying transactions, to make sure the system is protected.

Handling Test Results

Some of the results of the system tests will indicate that there are problems with the system. The test results that uncover problems should become system modification requests. Depending on the sophistication of the testers, you may or may not want to give the testers the authority to approve a system modification request for action before the new system goes into production. Only problems that prevent the system from going into production should be acted upon.

The most important point to remember is that only those requests that would prevent the system from going into production should be approved. It is easy to start making dozens of trivial formatting or data validation changes to each module that do not substantively affect the ability of the system to support the needs of the organization.

Determining Who Performs Testing

Testing can be done by people who do not have particular technical expertise. In fact, nontechnically oriented people are frequently better than systems professionals. Not only are they less expensive to hire, but they also better approximate the user population. In addition, automated testing tools are available that can help with many aspects of the testing process.

User Acceptance Tests

User acceptance testing involves putting the application in front of the users and making sure the users are satisfied. It also includes testing the documentation and training available. User acceptance testing helps identify any glitches in the user interface and inadequacies in GUI design. User acceptance testing should not be performed until the system testing is mostly complete and you are confident that the system is basically clean. The goal of user testing is not only to find any problems in the system; it also serves to generate user excitement. If the system has significant and obvious flaws, users will lose confidence in the quality of the system.

Performing small Pilot Lab Tests

Small pilot lab tests involve bringing in a small number of users and extensively training them on the new system. The strategy here is largely the same as in transaction flow tests: the users should do real work on the system in a controlled environment. Users can identify problems and provide feedback.

Training and Documentation

The review of documentation, training, and help desk functions should first be done internally within the IS department. It is far too common to assign a technical writer to build a manual and then start making hundreds of copies before anyone reads it. After the IS review is complete, show the documentation to a small group of users for feedback. Also get a small group of users to provide feedback on training sessions and the help desk system. Then repeat these tests with a larger group as part of initial user training.

Good-quality documentation and training are critical factors in user acceptance of the new system. Testing this vital portion of the system should not be overlooked.

Conducting Training sessions

At this point, if the system has passed all other tests and is close to implementation, users can begin real training on the new system prior to the Implementation phase. The first group of users brought in for training should be carefully observed. In addition, users should do real work to provide a full test run of the new system while they are under observation.

Modifications for Smaller Systems

For small systems, if the development team has a close enough relationship with the new system users, you can employ users as testers in the Test phase. This approach can also be taken if other

testers are unavailable. However, users may not test as thoroughly as dedicated testers. What is required are individuals who will click every button, try every function, try every combination of buttons and functions, and find as many bugs as possible. Waiting until the system is in production to find bugs means those bugs might not show up for weeks or even months—certainly long after the people who developed the code are gone. Even if developers are still around, they may well have forgotten the intricacies of the code.

Of course, most of the process auditing, verifying of the strategy and requirements documents, and so on is irrelevant for small systems. Testing small systems may merely consist of making sure users are happy with the new system.

For medium-size systems, careful verification of the early phases is not as important as for large systems. However, all system and user acceptance testing should be performed as described in this chapter.

The deliverables for small and medium systems are just as important as those for large systems. The reason that testing is less imperative for small and medium systems is that the time frame and number of system elements associated with the projects is small enough that there is a greater probability of doing it right the first time, perhaps making the final reaudit unnecessary. However, you may want to take small and medium systems through the entire Test phase for one of the following reasons:

- You have less confidence in the quality of the process due to test failures.
- The system is mission-critical for the organization. A higher level of QA is desirable.
- Organizational policies and procedures mandate a quality review of the entire system upon completion.

When Is the Test Phase Complete?

The key to completing the Test phase is to not try to fix every little problem that comes up. As many as possible of the system modification requests generated by the Test phase should be shifted to version 2 of the system.

To get the new system up and running, it is often necessary to "just say no" to more system modification requests. The system is complete when all type A bugs have been fixed. Type A bugs are bugs that prevent the system from going into production. For example, bugs that corrupt the data or prevent basic system functionality are type A. Type B bugs represent a failure to meet a system requirement. Examples include inadequate performance or a bug in a noncritical system feature. Type C bugs represent previously undiscovered system requirements that would significantly improve the effectiveness of the system. Type D bugs are any other desirable modifications to the system. Whether type B bugs will be fixed for the first system production release or not is a decision to be made by the project leader. Type C and D bug fixes are typically deferred to later versions of the system.

Before the Test phase is complete, the Strategy Document, Requirements Document, and design book should be validated for correctness. Either amend or document any differences between the delivered system and the system proposed in the Strategy Document. This report must be signed by both the senior project leaders and the chief user.

System and user testing, documentation review, and training are complete when you have taken care of all type A bugs. Anything that is not a type A bug is deferred until version 2 of the system.

Implementation

Once the system is complete, it must be implemented and maintained. The concepts associated with implementation and maintenance are quite different, so each topic will be discussed separately.

Overview of the Implementation Phase

One of the most important factors in a successful implementation is adequate user support. If you roll out a system to an untrained, confused, or resistant user population, the system, no matter how good, is doomed to failure. Providing high-quality user training and documentation, maintaining positive public relations, and managing user expectations are frequently overlooked steps. Reducing user apprehension, fear, or distrust of a new system, however, cannot be done entirely during the Implementation phase.

The best way to overcome user fear or resistance is to adopt an approach that keeps users involved in the entire system development process. Users must believe that the system is being built to their specifications to meet their needs and improve their jobs. Ideally, the system belongs to the users. The absence of user involvement often results in resistance, which can even manifest itself in sabotage. A number of years ago, the U.S. Postal Service attempted to implement a system without the support of the user community, and users went so far as to "accidentally" destroy electronic data-gathering devices. Granted, usually user resistance to new systems is not so blatantly destructive, but a resistant user community makes system failure inevitable.

"Another key to successful implementation is the ability of the project team to migrate the new application to the production environment. This process involves a transition period where the system is closely monitored and support is transferred from the project team to the system operation group. You should have chosen an implementation strategy at the end of the Analysis phase: either to use phased implementation or to implement the entire system all at once (the "big bang" approach). Your strategy also should specify whether to keep the legacy system running in parallel with the new system for a trial period. It is important that you make these decisions before the Implementation phase begins."(6)

Implementing the Entire System at Once

"If the 'big bang' approach is used, developers have to be very sure of the system. Any systematic bug is going to be multiplied by the amount of load on the system. Even trivial errors that don't show up in a controlled testing environment can become catastrophic when the big bang approach is used. The only rational justification for using a big bang approach is that the architecture of the new system demands it. For example, in a manufacturing environment, replacing one portion of an automated manufacturing process may not be feasible.

Frequently, the big bang approach is chosen for political reasons. It certainly can be cheaper, although much more dangerous, to implement the entire system all at once. Also, if the system is going to succeed, it will succeed much faster with this approach. There are certainly benefits to a big bang implementation. However, the question remains: Is a big bang approach worth the risk? Big bang implementations are very risky. You need to recognize that there is a material probability that the implementation will fail. Therefore, complete implementation failure would have to be planned for. You should not burn all of the bridges of the old system, since you might have to bring it back up in the event of a big bang failure. With the cost of an implementation failure being fairly low, there would seem to be very few cases where a true big bang implementation is a viable approach."(6)

Phasing in the System

If phased implementation is used, you need to decide how the new system will be phased in. The first option for phasing is on a functional basis. If an application contains several functionally distinct groups of modules (for example, sales, purchasing, and human resources), then it may be possible to isolate those module groups and implement them individually. The second option is to phase in the new system organizationally, perhaps implementing the system for only a portion of the organization, prior to rolling it out to all of the users. Phased implementation can be performed on a functional basis or an organizational basis, or both of these options can be used and the new system can be implemented a portion at a time to a subset of the user community.

The longer the phase-in process, the more expensive the Implementation phase becomes, but the

lower the risk. Whatever implementation strategy is employed should be carefully thought out and have the support of the user community prior to its initiation.

Paralleling

Another important decision made prior to the Implementation phase is whether to keep the legacy system running in parallel with the new system. Paralleling is a very expensive strategy. It requires all data entry to be performed twice. Users will have a sense that half of the work they are doing is a waste of time. If parallel implementation is chosen, it will be necessary to go to some lengths to support and help users get through this process. Hiring temporary help to reduce some of the burden of duplicate data entry on the users may be a wise investment.

If the organization cannot take the risk of any even temporary system shut down or failure, then paralleling may be the only option. Paralleling is frequently combined with a phased implementation. Also, it may be possible to parallel only key areas until the new system is performing adequately. Developers may want to consider using a parallel implementation for a longer time with a small subset of the organization followed by a parallel implementation of the full system for the entire organization for a short period (one or two days) to verify that the new system can run with full production loads.

Handling Implementation Problems

Problems that arise in the Implementation phase should be handled in the same way as problems in the Test phase: documented, analyzed, prioritized, and resolved. System modification requests should be used to document problems and enhancements. Any serious problems may necessitate aborting implementation until the problems are corrected.

Implementation Smaller Systems

Modifications for Implementation issues of small and medium-size systems are not greatly different from the issues associated with implementation of large systems. You still need to pay close attention to the effect of the implementation on the perception of the users. Flawed small- and medium-size systems can harm organizations just as much as flawed large systems. However, because a smaller system is easier to test, you are more likely to be able to successfully implement a small system using a big bang approach.

If a phased implementation is necessary for a small system, the duration of the implementation period will likely be shorter and the implementation plan will be simpler than that of a large system. One of the great dangers of small systems is to underestimate the importance of a careful, well-thoughtout implementation phase.

For medium-size systems, you should use the same strategy as for large systems. For these systems, a shorter phase-in period is often used.

It is easier to support the users of small- and medium-size systems. For these systems, running the legacy system in parallel for a limited period of time may be appropriate.

When Is the Implementation Phase Complete?

Implementation is complete when the following three criteria have been met:

- In phased implementation, the phasing has been completed and implementation has been done throughout the entire organization or whatever portion of the organization is appropriate for that application.
- All parallel processing with the legacy system has been discontinued.
A suitable period of monitoring has passed that allows you to be confident that the new system is up and running.

The last step is the most critical. Significant effort should be devoted to monitoring system performance, system resource usages, and user perceptions for the first few weeks after implementation. It is only after this intensive monitoring period is over that you can declare the Implementation phase complete.

Maintenance

Maintenance is a process that continues throughout the life of the system. A predefined process must be put into place to handle ongoing problems, modifications, and enhancements.

“One interesting aspect of maintenance is the way requests are viewed. Many maintenance requests are not necessarily a bad thing. Some companies use the number of maintenance requests as a primary measure of system success or failure. Some companies may consider a large number of

maintenance requests as an indication that the system is full of bugs and not meeting user requirements or expectations. Other companies may view a large maintenance queue as a sign that users want to fully exploit the system's potential.”(6)

Earlier chapters discussed how users log requests and problems through the system modification request process. In the Maintenance phase, these requests are ranked and decisions made as to when they will be addressed, if at all, and assigned to an appropriate version of the system.

Versioning

Changes to production systems should never be made on an ad hoc basis. Modifications need to be prioritized, grouped, approved, applied to the system, and then thoroughly tested before the next version of the system is implemented. The approval and prioritization process should be handled by a steering committee made up of both developers and users. When requests are approved, they should be assigned to an upcoming version of the system to indicate when these modifications will be implemented. After all of the scheduled modifications for a version have been applied, the new version must be tested prior to implementation.

Of course, certain system modification requests cannot wait for a new version and must be implemented immediately. For example, a bug that corrupts underlying data cannot wait for a new software version but must go through an immediate testing and implementation process.

Version Testing

Version testing need not be as comprehensive as the original testing of the system. You are making changes to an already thoroughly tested system, so you do not need to repeat the full testing process for each new version release. However, you do need to perform incremental unit testing for each affected module. Not only should the changed portions of the module be checked, but a full retest of any affected module should be performed unless the modification was cosmetic.

In addition, all integration tests associated with all affected modules should be performed again, as should transaction flow tests for each physical process flow that includes an affected module. Since this testing process is the same no matter how many changes are made to a module, it is much more cost effective to group as many system modification requests together as possible and implement them together.

Modifications Necessitated by Software Product Changes

When the underlying application software, DBMS, operating system, or platform changes—for example, when a new Oracle Developer version or the next version of any software is released—you must not assume that the system will still operate correctly. Whenever you want to change to a new product or version, the transaction flow and stress tests should be repeated prior to implementation to ensure that the system operates correctly with the new architecture. Coexistence testing is also necessary if new systems that interface with the existing system are brought online.

Maintenance Modifications for Smaller Systems

For small systems, versioning may not be necessary in the early phases of the project until the system is stable. After the initial Implementation phase, small systems should be versioned just like large systems.

For medium-size systems, maintenance should be handled just as for large systems.

When Is the Maintenance Phase Complete?

Of course, the flippant answer to this question is that Maintenance is never complete. However, the reality is that systems do have a finite lifetime. One of the most difficult questions to answer is: when is it time to stop all but essential modifications to the system and begin designing its replacement? There is an economically rational way to make this decision. You can carefully analyze the amount of money you are spending on maintenance versus the amount of money it would take to develop a new system. Of course, when you build a new system, you have the opportunity to reengineer the business processes. In practice, systems frequently stay in production long after it is economically rational to keep them in production. Maintaining legacy systems remains a fixed, if not increasing, cost over time. As the business grows and changes, the ability of the system to keep pace with those changes decreases.

Systems are usually only replaced when failure to do so would cause great distress to the organization.

Conclusion

Implementation marks the last of the CASE Application Development Method phases. However, once the system is complete, the developer's work is not finished. No system can succeed without competent maintenance. The ongoing satisfaction of the users depends upon how well the system can respond to requests for changes.

Following the CADM process does not guarantee that a project will proceed completely smoothly or that it will be on time or under budget. However, if you follow the CADM process throughout system development, you can be assured that your system requirements are coherently gathered and that you can track any system requirement from its point of origin to its eventual system impact, if any. All of the audit and quality control portions of the CADM process virtually guarantee that the final system faithfully fulfills the system requirements as the developers understand them. By carefully following the methods outlined here, you will have sufficient documentation of the process to be able to defend your decisions throughout the process.

All of this does not guarantee that you will build a good system; however, CADM does guarantee that you will build the best system that can be generated by the development team. Good systems require not just a sound development process, but also skilled, creative developers.

Chapter 5 : Experiences in developing IS in the world

When we started working on the project of developing an IS in the General Hospital of Zadar, we had a look at similar experiences in the world. Here is a resume of some of the initiatives in the world that inspired us.

Building a national information system in the UK

The Information Authority is part of their National Health System (NHS), and their role is to develop IS projects and coordinate their implementation. It is a good example for other countries. If the Ministry of health does not have a department for developing IS on the national level, we shall probably see disorganized development on that level. A lot has been done, and what is most interesting is that the documents about these projects and their implementation are available on the Internet. It is useful to visit these sites because everyone can learn a lot about working on such projects.

<http://www.nhsia.nhs.uk>

<http://www.nhsia.nhs.uk/def/home.asp>

You can find the information about:

- data models used in IS. If you remember from the previous chapter, building a data model is very important for defining the standards and framework for building applications. A description of entities and their attributes can be found at <http://www.nhsia.nhs.uk/datastandards/pages/ddm/index.htm>
- the concept of a National electronic library for health, where physicians, nurses and patients can find recent information. It is maintained on the national level.
- an initiative to evaluate hospitals and teams. The results of this evaluation are public. So you can always see which hospitals and which teams are highly rated and ask your GP to send you there if you are ill.

The European Union (EU) approach

Since the EU consist of many countries, the need for a 'national' or 'super-national' level approach is even more emphasized. There are lot of institutions and committees who are involved in the process of the development of standards and initiatives for coping with the diversity in these countries. One such committee is the European Committee for Standardization or Comite Europeen de Normalization (CEN). This committee has a subcommittee called Technical Committee (TC) 251 which deals with problems in health informatics. In the literature it is usually referred to as CEN/TC251. They have meetings and working sessions, and there are a lot of documents describing various initiatives and recommendations. All the documents are downloadable from the site <http://www.centc251.org> There are recommendations about the purchase of storage, databases, and operating systems for the member countries of the EU. You can find it in the document http://europa.eu.int/comm/di/pubs/arch/architecture8_en.pdf.

It is logical that if you want to build a transparent information system in all member states of EU, you must work out some standards and give recommendations. There is a recommendation of the Unix/Linux platform and Oracle as the database. Oracle is portable on all platforms, Windows, Linux, Unix, VMS.

The Netherlands Leiden project (HIScom)

We could say that this was one of the first attempts to build a kind of national health information system. It started in 1975 and is still developing. It is used by most university hospitals in the Netherlands. Now HIScom is one of the leading IT companies in the Netherland.

<http://www.hiscom.com>

Diogene HIS in Geneve Switzerland

As in the above-mentioned example, the University Hospital in Geneve started to develop an integrated information system. It started in 1971 and was implemented in 1978. These early systems can be a very good source for both good and bad ideas. A lot can be learnt from other people's mistakes.

<http://www.uhc-ge.ch>

The Cleveland Clinic Foundation

These comments are based on what I saw while visiting this hospital during a period of IS reengineering as well as written materials about their implementation of a new IS. Like many other hospitals in the US, the Clinic started implementing information technology in health care very early. The policy was to buy applications which were considered the best on the market. For example if some company offered a good program for the laboratory you bought it. If another company offered a good program for radiology, you bought it. Step by step the hospital was full of various applications, built in various program languages, various user interfaces and even using various operating systems. This policy is called 'best-of-breed'. But the problems with compatibility, communication, exchange of data were so big that the number of IT specialists to maintain those systems is very big, too, and it became very expensive. There we might be able to see the origin of the idea to build programs used for data exchange between applications, like HL7 or Edifact.

With the development of new technologies in hardware and software, the management of the Clinic started with process of rebuilding their information system.

One important fact I learned there, and it was part of their experience, is that many projects of this kind fail mostly because of ineffective implementation. Some projects have the right solutions, but fail because of implementation issues. Project managers must bear this fact in mind all the time. And among other things, the single biggest cause of failure is the **inadequate preparation of the workforce for change.**

Some ideas, methodology, and plans you can find in article:

Memel at all.: Development and Implementation of an Information Management and Information Technology Strategy for Improving Healthcare Services: A Case Study

Journal of Healthcare Information Management (JHIM) 2001, Volume 15, number 3, p.261

Experiences in Croatia

Initiative of Ministry of Health

Like many other countries in transition, Croatia had no documents and plans for building a health care information system. A 2 years ago, as an initiative from the Ministry of health, a task force was formed to do this job. In the fall of 2002, an international invitation for tender was announced in the newspapers. This was a document that specified what information systems for a hospital must contain. The bidding has been closed, and we do not currently have any information when or whether this project will start.

There are a few companies that have submitted their applications, with prices for licenses from 70 to 100 million euros. One of the conditions in the bidding document was that companies must implement this system in one of our clinics within one year, and after that the Ministry and government will decide what the best application is and sign a contract. No decision has been made till now, and change of government this year, made this process even slower.

Drawbacks of this document and processes are as follows:

- There is no national level data model. It means we must accept data models from other countries. Sometimes there are problems such as legislative, billing, and standards compatibility.
- There is no strategic document about what the national level information system should look like. For example, will every hospital have its own hardware and software, or will there be data centers in the regions? There is no unique identifier of the patient on the governmental level. Without that there is no CPR, and there is no exchange of data between various subjects in health care.

The future will show what will happen. Bearing in mind the experiences of other countries, like Romania, there is a lot of work to be done. As the experience of the World Bank, which finances big infrastructure development projects like these in transition countries, shows, the implementation of highly sophisticated packages often does not give results. Lack of organization, knowledge, and experience sometimes produces very bad results.

Experiences in the General Hospital of Zadar, Croatia. (GHZ)

Building an IS in GHZ started in 1988 with a mainframe computer and 10 terminals. The general idea was to build an IS for all subjects in health care in Zadar. The only solution was the mainframe computer because in 1988 PC technology and networking were at the very beginning of their development.

We had a team for building applications ourselves, because there was nothing available to buy. The period of war in Croatia from 1991 to 1995 slowed down this initiative. In the meantime technology changed a lot and we started reengineering our concept. We stopped putting our efforts into developing new applications on the mainframe. We maintained the existing applications, but started planning how to substitute the old system with a new one. Although it was very hard to work on two fronts, we made a project and presented it to our board. During 2002 we established a LAN in the hospital. We now have about 400 connections. Every room where someone can work has a connection to the LAN.

When we learned of this current initiative from the government, it put us in doubt about what to do. Should we wait until the government gives us their solution or develop one on our own. An intensive discussion followed and eventually we decided as follows:

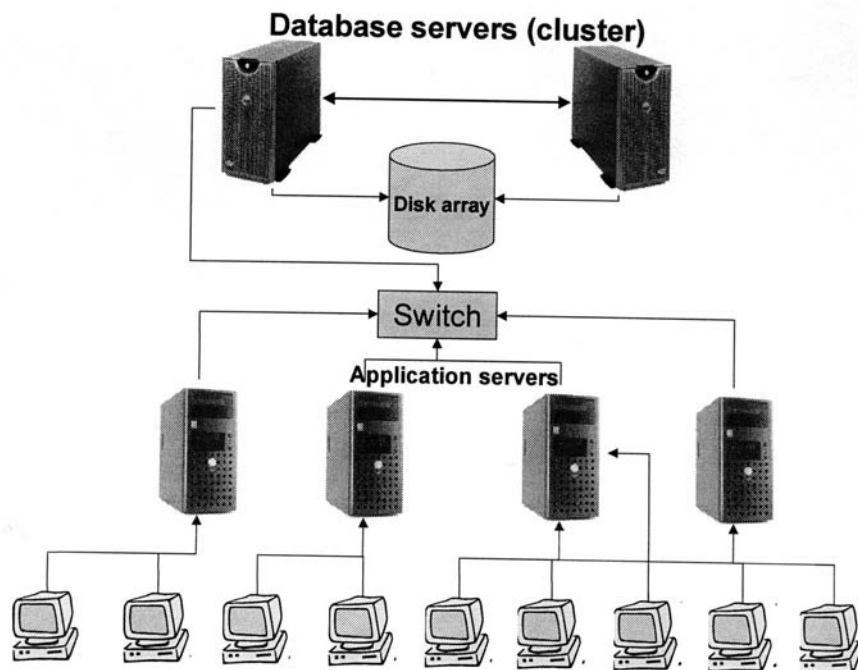
- We have something and have to replace it
- If we wait for a few years (the estimated time for the implementation in all hospitals is 4-6 years), what shall we do if our system crashes without a possibility for recovery?
- We should build our system because even if we spend money on equipment, licenses and education, it is not wasted money. We work and learn and people are used to new systems of work and new technology. After a few years we would think about replacing servers, anyway. So we can't lose.

We made an invitation for bidding of equipment during July of 2003 and installed all equipment during November of 2003. After we installed all necessary software, a testing phase was initiated in January 2004. We have prepared the first phase of the project: working with the ambulatory care and diagnostic departments and the pharmacy. Work with hospital care (inpatients) will start next year. The development of a hospital information portal and digital radiography will be implemented in the third phase. After all phases are completed, we will have about 350 clients and 150 printers installed. We decided:

- to use Linux on servers. One study of Aberdeen group says that Linux will be installed on 50% of servers in the world. Open license and stability are two of the most important reasons.
- database servers will be cluster servers empowered with Oracle RAC. This will enable 24/7 functioning of the system, which is very important for healthcare systems.
- to implement a three-tier architecture with thin clients on the user end. This decision was based on some experiences in Europe and the US. The cost of ownership is much lower for thin clients than for standard PCs, and the maintenance of thin clients is much easier. There are no moving parts which cause failures. Most programs and user rights are defined on the server, so it is easier to maintain it. Changes in PC technology and the operating system you must license are much faster than in thin clients. On the other end there are five application servers. If one server fails, the others will take the users from that server. This is called "fail safe." It is good for load balancing too. You can redirect users from a heavy loaded server to a less loaded server. (see figure 3.)
- to rely on Oracle as our database and the tools that go with it. Although it is an expensive solution, Oracle is the only database that works on almost all platforms (all operating systems) and the only solution that provides an application cluster which works 24/7.

We decided to use PC technology instead of the mainframe because it is too expensive for us. One estimate is that these two phases will cost us about 700.000 euros.

The training of users will be of hierarchical type, such as "train the trainer." A number of experienced users will be trained and they will train the others in the departments. About 400 users are to be trained. If you organize education for every user, it would take too much time. So it is better to spread the process of training between more trainers. A special training center will be organized for this purpose.



3. Figure Three-tier architecture

All this is made possible only through the significant efforts of the development team and the vision of hospital management and staff.

References:

1. Abdelhak, M., ed., Health Information: Management of Strategic Resource Philadelphia, W.B. Saunders Company, 1996
2. van Bommel, J.H., Musen, M.A., Handbook of Medical Informatics, Heidelberg, Springer-Verlag, 1997
3. Memel, D.S., et al.: Development and Implementation of an Information Management and Information Technology Strategy for Improving Healthcare Services: A Case Study Journal of Healthcare Information Management (JHIM) 2001, Volume 15, number 3, p.261
4. Van de Velde R.: Framework for clinical information system. Int J Med Inf. 2000 Jan;57(1):57-72
5. Heitman KU et al.: Calming admins and managers – experiences from four years using thin client technology in a hospital environment. Stud Health Technol Inform 2000; 77 :912-5.
6. Koletzke P., Dorsey P., ORACLE Designer handbook, Oracle Press 1999.
7. <http://www.dulcian.com/papers/CADM&CDM.htm>