Specifying System Requirements

Overview and Objectives

As you probably have noticed, we live in an information age. The World Wide Web has truly revolutionized the way in which we access information and provide information. From hotel and airline ticket reservations, to on-line banking, long-distance education, almost everything now is driven by information flows among us and the systems that are accessible either via the Internet or within the local intranets at our workplaces. Healthcare is no exception. A significant component of health services is information flow, among patients, clinicians, administrators, health devices and so on.

Underlying our rich and powerful set of technologies are "databases". So clearly learning about databases is not just interesting but essential if we are going to advance in our jobs and become more efficient knowledge workers. The purpose of this course is to teach you the methodology and procedures that will allow you to design these very important constructs, i.e., databases. This course has 3 components:

1. Abstract business process into system requirements,
2. Model system requirements into a database, and
3. Use Standard Query Language to gain access to the data.

This lecture belongs to the first component of the course, i.e. abstracting business processes into database requirements.

Design of anything, including a database, is both an art and a science. As a database designer you have to make many important choices about what information you want to include and what you want to exclude. What you include will change the future focus of the organization and what you exclude may haunt the organization when important information they need is missing. Where you put the data and how you structure it will also matter. Because if data are not where others expect, they cannot find it. People can put the data into your database but cannot get it out -- how frustrating. So it is no surprise that there are a number of rules for how to design well.

A database keeps information within the fields. The design of the database is inherently tied to specification of the fields. This section describes how one can decide which fields should be included in the design of a database.

Glossary of Terms Used

Design of a database, by definition is the process of abstracting business processes into an artificial set of concepts and ideas. It is a process fitting circular pegs into square holes - obviously not perfect. The reality we face is rich and complex and we have to abstract this reality into a new set of concepts and ideas, which by definition can not be as rich. To complicate matters more, we need names and definitions for these concepts and that introduces a lot of jargon. Its even worse when these names are abbreviated in acronyms, making it more difficult for the novice to understand what is intended. Then Database design is made more difficult than necessary. To simplify, this section introduces a minimum set of new terms that we encourage you to understand and keep a mental picture of it as you proceed in designing databases:

1. An information system includes one or more databases, data interfaces and automated processes. Modern information systems encompass not just the database but all the interfaces, such as the web pages we are familiar with, as well as the software that manages the interactions of the users with the system and provides the business logic of the overall system.
2. The goal of database design is to express reality in terms of tables (fields and records) of data and the relationships among the tables.
3. Tables maintain data on objects, people or events. It consists of columns and rows.

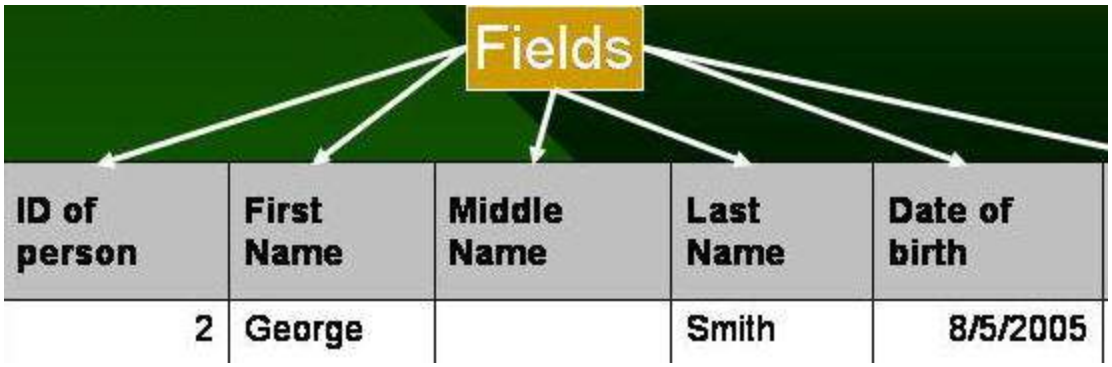

Figure 1: Definition of Fields

4. Tables contain "fields or attributes." Fields or attributes provide the label for the data stored inside tables. Visually they are the columns in the table.

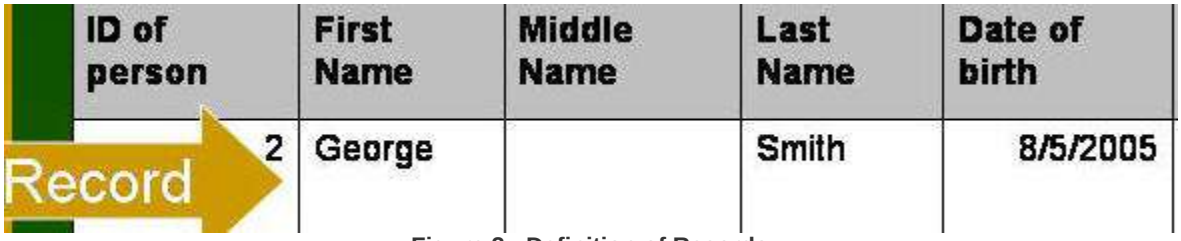

Figure 2: Definition of Records

5. Tables also contain records. Records are a collection of data representing a unique instance of the table. Visually they represent the row in the table.
6. Relational databases are based on relationships among tables. A relationship is one or more shared fields between two tables and represents a connection among objects, people or events. A database may capture many different relationships: Patients have physicians of record. There is a relationship between a drug and its manufacturers. We are related to other individuals in a variety of ways, for example, by family relationships, by business relationships, by work-related relationships. A database must capture all those relationships that pertain to its purpose.
7. Actors are humans or systems who interact with the database. A database is designed so that actors can decide. Actors also provide data to and receive information from the database.
8. A decision is a choice between at least two alternatives. A datum is included in the database if it is relevant to the decision, i.e. it helps choose one alternative over another.
9. A scenario is the way in which an actor's involvement in a decision leads to changes in the database. Some provide input and others make decisions based on the database output. A scenario describes the information flow between the database and the actor in the context of decisions addressed by the database.
10. A use case refers to the database behavior under a particular scenario. It describes what the actors see when they follow a scenario to interact with the database.

1. Establish the purpose of the Database

Design is a purposeful activity, in other words databases are designed to accomplish specific objectives. Whenever we engage in the process of designing databases we are implicitly dealing with a specific business domain. As individuals in the health care business, our business domain is likely to be any of the various aspects of health care. For example, if we are working in a hospital then the business domain may be all the activities that take place in a hospital, from patient care, to human resources to maintenance of the facilities.

Clearly, when we plan to design a database, we need to narrow the scope of work to something practical. As health care professionals, the scope of our work is narrowed to the business and clinical domains of the organization. Since these domains could be immensely large, we need to carve out a more specific purpose for the database. A good designer always asks what is the purpose of the database: "What is the point?" Unfortunately, the answer to this

question is not always clear. Different people want the database to accomplish different tasks. In the end, asking "why" helps reduce the task that should be modeled into a database but does not solve the design problem entirely.

Establishing the purpose of a database is not simple. Different actors express different purposes. For example, we were asked to design a database for a mental health court. A mental health court diverted mentally ill but not dangerous patients from the usual court procedure into a special procedure designed to reduce their return to the criminal justice system. The program is called Forensic Alternative Services Team (FAST) and involves a number of social workers and other clinicians deciding on the eligibility of the case for the program and monitoring the clients to see if they comply with court ordered treatment plan. The initial request for help came from the mental health court judge who wanted a better understanding of the effectiveness of her court. This was one purpose for the project. The mental health community agency funded the pilot project and the lead person in this community agency wanted a simple database tracking the performance of the FAST program. This was a different purpose. The funds for the pilot were provided by the mental health agency to the University because of impending collaboration between the court and the University on a special project. The university personnel wanted a database that would use standardized questionnaire that could be used in national program evaluations. This was still another purpose. Discussions with the FAST program director revealed that a medical record system was set up for them but had gone unused because they did not want such details. They wanted to stay as much as possible with the existing paper record and at most to have a case management system and not a medical record. This was still another purpose entirely. All of this was occurring in the context of the Federal government encouraging the development of Regional Health Information Networks designed to automatically integrate data across organizations, enabling the court, the treatment providers and the FAST team share data automatically. The more you talk to organizational leaders the more you understand that there are multiple purposes for a database, some of which the designer can meet and others for which the designer cannot meet immediately but can anticipate so that a future expansion was possible.

2. Analyze What Exists

Database designer can understand a great deal of organization's needs by examining current operations. They can note the tasks that are being done in the organization, the paper flow. They may even be able to review existing information systems -- if they exists. But the information that emerges from review of existing conditions are suspect because they do not involve the client and because they do not reflect future needs. Take for example the possibility of assessing information needs by examining the tasks at hand. The analyst may observe the organization tasks and the information items needed in each. But organizations are complex and learning how information flows in them is time consuming. An analyst needs to spend weeks becoming an expert in the organization's information flow and even then they are no more of an expert than the organizational leaders who breath and live in the environment. Why spend so much time studying a situation when one could easily ask the user -- who is the real expert on their own needs?

Of course the ideal situation is to do both: examine the existing database and ask the organizational leaders. Examining the current conditions make sense as a starting point but soon we need to move beyond this level of analysis and involve information users. Examining the current situation has two components. First it is important to see the existing databases and their structures, no matter how inconsequential they are. Often data are kept in flat and inflexible files. For example, when we were asked to design a database for a mental health court, we found that they had a relatively flat database with six tables. These tables closely corresponded to forms they had.

Field name	Field name	Field name	Field name
Client ID	School number	Discharge status	Relation
Program	School type	Admission time	Relation
Site	involved	Admission type	Ref required
Program group	DSS involved	Referred to	Ref required
Admission date	Custodian	Hospitalization	Co-pay
Discharge date	Homeless	Plan code	Co-pay
Primary staff /team	Foster care	Pay source	Update date
Presenting problem	Foster care placement	Pay source	Entry date
Referred from	Furlough	Payer 1	User name
Target population	Crisis bed requested	Payer 2	
Axis I	Last ITP date	Insurance type 1	

Axis I	Treatment plan date	Insurance type 2
Axis II	Dr order date	Policy number 1
Axis II	Last assessment date	Policy number 2
Any Axis III	Discharge type	Effective 1
Axis IV	Hospital discharge date	Effective 2
Axis V / GAF score	Final agreement date	Expiration 1
Legal problem	First contact	Expiration 2
Alcohol problem	Referral date	Policy holder 1
Drug problem	Discharge referral	Policy holder 2

Table 1: List of Fields in One of the Tables in the Existing Database **More►**

The tables in the existing database kept multiple pieces of unrelated information in the same place. For example, it put name, address, demographic, arrest history and medical history in the same table. Clients of the court changed their address frequently. Every time the address changed the entire record needed to be changed. Every time the client was arrested, another entire record needed to be put in. The value of examining existing databases is that they suggest potential fields and may explain why existing databases are frustrating. In this manner, future design can avoid past errors.

Second, it is important to trace the paper flow. Many organizations have paper forms that document decisions made by them. These paper flows are an important source of ideas regarding what information might be kept in the new database. For example in the mental health court example, the following paper forms were used:

1. The pre-admission screening form **More►**
2. Demographic form **More►**
3. The admission form **More►**
4. The monitoring form **More►**
5. The discharge form **More►**

The forms pointed to several decisions that the organization needed to make. The form on pre-admission screening and the admission forms pointed to the decision about diverting a defendant from usual jail and court processes to special processes set up for mentally ill patients. The demographic and the monitoring form pointed to the information needed for deciding on the effectiveness of the diversion process. The discharge form was used to decide about court staff's assigned work load. These forms point to what data might be needed for which decisions.

3. Identify Future Decisions

One obvious way for assessing information needs is to ask organizational leaders what should be included in the database. Information users have a great deal of problems in articulating their own needs. When asked, they often produce a long wish list of data that those not correspond to their true needs. They include items that they will not use and do not mention items that they will need. Organizational leaders fall victim to their own cognitive limitations. When interviewed, they have to remember how key information was missing in the past decisions. This is very unpleasant. No one wants to remember failures or episodes in which their needs were not satisfied. Unpleasant events are often forgotten, especially when you, yourself, are in charge. Interviewing organizational leaders about failures of their organizations is akin to asking them to fess up to their own mistakes. It is an unpleasant task that many rather forget. Furthermore, many decision makers do not realize how their own needs change over time and how these needs are affected by external events. They are so engaged in everyday work that they have not had an opportunity to envision a different arrangement. Many are not aware of new technological possibilities and keep projecting future needs based on their existing expectations. In short, many fail to imagine a new future. To overcome these limitations, it is important to put organizational leaders in a new mind set. The analyst can do so by first ask what are future decisions that the database should address and then assess information needs within these decisions. Clearly, information needs change over time and by repeating what has been to date, organizations may put themselves in a disadvantage. This may be acceptable in a stable environment, but not in a dynamic one. A focus on future decisions transcends this pitfall and provides a context for the value of various pieces of information.

Information collected in a databases data that have been consistently relevant to a particular decision. This is how an analyst can decide to collect or to ignore a datum. If in a given decision, there is no requirement to examine a price

of data before making the decision then the database should not contain it. Only that portion of the information which we will use need to be stored and retrieved at a later time. Other information may reside elsewhere in the system, may be relevant to decisions not addressed by the database or may be handled in a different way at the time of decision making. A focus on decisions makes sure that long wish lists of information items that are never used do not clutter the system and drawn out the opportunity for collecting what really would influence actors.

An analyst has no choice but to select a few fields and focus on them rather than all possible fields of information. If you look around you, you'll see many objects. Each one of those objects has potentially hundreds, or even thousands characteristics. For example the monitor in your computer has a model number, a type, a model name, a type of display, a color or colors, a height, a width, a depth. Is that all? Well, probably not. It depends on how detailed you want to be. You could also describe the lettering that the monitor may have, where that lettering is located, what color is the lettering in. Have we exhausted all the things we can say about the monitor? Hardly. We could take a magnifying glass and look for scratches and indentations on the frame of the screen, we could describe what the chemical composition of each one of the parts of the screen is, and so on and so forth. Clearly, objects are awfully rich in terms of their characteristics. An analyst must figure out what is pertinent for the database purposes and what is not. If I am a clinician I will be interested in those characteristics of a patient that are pertinent to my decisions about his health. If I am a banker, I will be interested in the financial characteristics of my client. So for every domain there are objects that are pertinent, and for each one of those pertinent objects there are characteristics that support the purpose of our database. Everything else is irrelevant and we must filter that out when considering how to build an information model. Since data become informative in the context of particular decisions, then the simplest step is to evaluate the relevance of a datum in the context of a decision that needs to be addressed by the actors.

By focusing on future decisions, the databases designer anticipates various roles the information system might play. For example, in the mental health court project, we were given a tour of the court. While at the court, it became clear that many defendants come to the court without their data being available. A significant component of the time is spent in the court sorting out who knows what about the defendant. In one case, the FAST social worker reported that they had not been able to get a response from the health care provider regarding defendant's competency examination. The probation officer reported that he has not been able to locate the client in the community. The defense attorney reported that he had not met with the client yet to see how the examination went. All along the judge was left waiting for information. Eventually, the judged called the doctor involved and talked to him on the phone who collaborated the client that that she had gone to the clinic, had been examined but did not want to take the medications prescribed. It was clear from this case that an exchange of information between the FAST database and the clinic would solve the problem. Health care providers, for example those in the Veterans Administration, were reluctant to provide the care because of extensive court required paperwork. With the automated exchange between the clinic databases, the court will be well informed and the clinicians would not need to file additional forms. But the FAST social workers seemed unaware that the Veterans Administration had a great electronic medical record system and that it was possible to create the electronic link to it. When the option was described to them, they were not sure that such links can be made. So asking the organization employees about what they need may not be enough. Sometimes, especially when it comes to automation of tasks previously done manually, designers of new databases need to lead the effort.

A similar example occurred when we noticed that FAST social workers, treatment programs and court staff have a difficult time identifying cases. Many patients give different names at time of arrest and in clinics -- creating a havoc in matching the information to the right person. One way to overcome this is to include an image of the person in the database. When employees were asked what they want in their database, none mentioned an image of the client. When we suggest that this could be done, they were very enthusiastic. The point is that designers cannot completely rely on organizational leaders on what should be included in the design. Part of their task is to push for change in business processes.

4. Invite a Panel of Experts to Create Decision Making Scenarios

To be effective, an analyst must ask both organizational leaders and outside experts to identify the needs of the organization. By relying on a group rather than an individual, the analyst minimizes the cognitive and behavioral limitations of having a single person define information needs. The information system that emerges may not fit the idiosyncratic needs of a particular individual but it transcends people who come and go and serves the organization as a whole -- no matter who is in charge and for how long. By adding external experts to the membership of the group, the analyst emphasizes what people in the organization want and what people outside think they may need. It brings an outside point of view and breaks up the set ways of doing things.

In the mental health court example, we had heard of a Federal system for probation officers. In some sense the FAST social workers were doing activities similar to the Federal probation officers: they were tasked with monitoring adherence with the court ordered actions in the community. To gain a better understanding of what the FAST database needed to look like, we examined the Federal system.

Scenarios are snapshots of a story you are telling about how a database will be used. You should make different scenarios about each decision. In the scenario you should talk about how the actors will interact with the database to make a specific decision. Here is an example scenario on admission to the FAST program:

> **Scenario short name**: Admission
> **Decision affected by this scenario**: Admission to FAST program
> **Actors**: Jane, Connie, and Sheila
> **Description:** Sheila enters the demographic and arrest records, Connie and Jane review the patient and enter presumed diagnosis and their recommendation for admission to FAST program. Through out the process the database maintains the information and assists in the communication between people involved.

Additional scenarios can be written about how the database helps in other decisions, for example the decision about what treatment should the patient participate in. In the end when you have developed a number of scenarios, you have a way of telling a complete story about what the database will do and what decision it will have an impact on.

5. Develop Use Cases

The use case method was pioneered by Ivar Jacobson, one of the three creators of the Unified Modeling Language, an important language in designing databases. The purpose of a use case is to describe the behavior of the system from the point of view of the user interacting with that system. By stating how the system reacts, that is, what it does, when the user gets in contact with the system it is possible to gather the kind of information that will be needed to support these system actions.

For example, in the mental health court system a receptionist was asked to collect information from the Federal and State databases regarding a client's "wrap" sheet. This information was then put into the computer to document the case for a client being considered for the FAST program. A form was provided to the receptionist asking her about the client's demographic and arrest information. This form was then communicated to the FAST social workers who established the client's presumed diagnosis and eligibility for the FAST program.

In Unified Modeling Language one uses a series of icons to graphically depict a use case diagram. The actor is always external to the system and is normally represented as a stick figure with a label underneath it that identifies what kind of actor it is. Note that the term "actor" is not meant to be restricted to human users alone. Any object in the business domain that interacts with the system can be considered an actor. For example, another system can be an actor, if it sends or receives information from our database.

The database or system is normally shown as a rectangle, and it also has a label that identifies it within the business domain. Inside the rectangle that represents the system there are ovals which correspond to the actual use cases. Each oval has a label which can be placed either inside the oval or immediately below it. The label is a short-hand for a given functionality of the system. The recommended practice is to use succinct phrases, for example verb-noun pairs, for the label of each use case to convey what the system does.

Lastly, straight lines are used to connect actors to the various use cases of the system. When an actor is connected to a use case in a system it indicates that the actor interacts in some fashion with that functionality of the system. Needless to say, the bulk of the description for what that interaction consists of is not shown in the diagram itself, but is captured in a descriptive narrative. The diagram is simply a very high level depiction of the actors and their connectivity to the system. The tools for Uniform Modeling Language normally offer data entry forms where this information can be documented. In absence of case tools one can use a word processing software to document a use case. By following this discipline we can get a handle on what type of information is required by the system in order to operate correctly.

The process of building use cases focuses on the message flows exchanged between the actors and the system. There are four main categories of actors which can be documented in a use case Diagram. First we have (a) the principal actors, which generally will be humans interacting with the system and its various functionalities, next are (b) the secondary actors, which normally are individuals performing administrative or maintenance functions with respect to the system. The third category of actors is (c) the external hardware. This includes those peripheral devices that enable the Input / Output functions for the system. For example, the terminals through which a human user can log in, the printer devices that generate hard copy reports, etc. Lastly, there are the (d) external systems which interact with our system. In a complex application our system may be linked to a local network which provides the connectivity to the Internet, or to the e-mail server within the organization.

Each of these potential actors may send to and receive information from the system, and each of these flows may depend on the sequence of interactions between the user and the system. A given sequence of interactions is normally bundled into what is called a scenario. A patient accessing an on-line medical system may participate in multiple scenarios depending on what sequence of interactions he engages in. In one scenario, the patient may be simply searching for answers to a query posted previously. In another scenario, the patient may be reporting his health status after having followed the medical treatment recommended by his physician of record. Different use cases can then be documented for each scenario. All scenarios must be linked to decisions. In the first scenario, the patient is deciding whether to come in for a visit. In the second scenario, the provider is deciding if the patient needs to return for a follow-up visit.

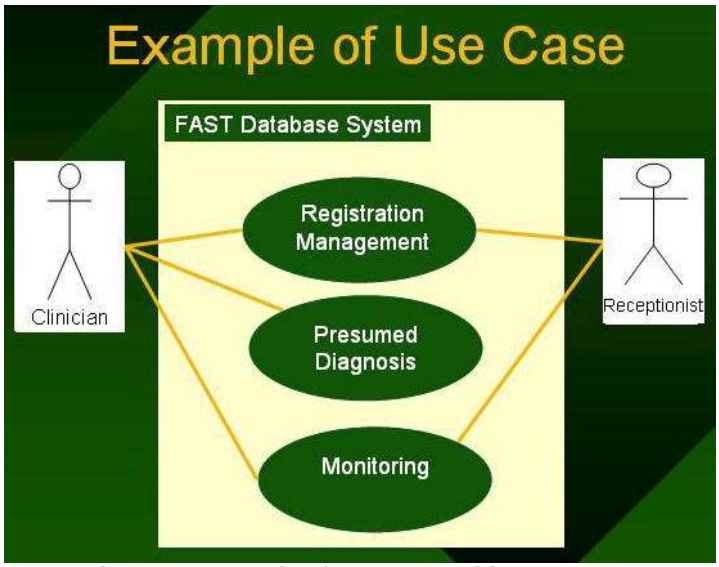

Figure 3: Example of use cases with two actors

The following shows an example of a use case Diagram using Uniform Modeling Language notations. The system we are dealing with is the mental health court system. A receptionist is shown with access to the registration use case. The FAST social workers are shown interacting with module for presumed diagnosis and monitoring. We have

at least two actors, namely, the receptionist and the FAST social worker. The functionality of the system for this scenario is shown in the three use cases shown in the diagram, namely, the Registration Management, the Presumed Diagnosis and the Monitoring component. The diagram shows that the FAST social worker interacts with all three of the system functionalities, whereas the receptionist is focused on the registration component. More careful examination of the system revealed several flaws in the use case scenarios. First, one of the social worker was disabled and could not interact with the computer directly. This necessitated the need to send information that needed to be put into the computer back to the receptionist. Therefore, in reality and at least for one social worker, the receptionist did all of the interaction with the system.

But pictures do not tell the entire story and we need to have the details captured in the use case document. It is generally recommended that a use case document should contain:

1. *The name of use case*
2. *The name of scenario which this use case belongs to*
3. *The beginning of the use case:* This is normally the event that triggers the start of the use case.
4. *The end of the use case:* This is normally the event that stops the use case.
5. *The exchanges of Information:* This is a description of the information flows that take place between the system and the actors. It is normally recommended to use a plain language style, such as, "The on-line patient fills in the form and provides his first name, last name, etc."
6. *Measurable result:* This section describes the impact of the use case on the database, e.g. creation of a new record, deletion of a record, etc.

Of course other information can also be kept about each use case:

7. *The chronology and the origin of the information:* Here it is documented when the system requires internal or external information and when it records it internally or externally.
8. *The interaction between the use case and the actors:* This portion of the documentation sets out the boundaries between what is inside the system and what is external to it.
9. *Behavior Repetitions:* A description of the system's iterative behavior. In most cases this is done using pseudo-code.
10. *Optional Situations:* Where multiple paths are offered at a certain point in the sequence of events one should list them, followed by the what the user may do next.

There is no canonical form that one must always adhere to for documenting a use case. Here is an alternate way of documenting the Registration Management use case.

Documenting Use Case

Overview
The purpose of this use case is to allow a receptionist to register into the system a client likely to benefit from FAST program

Primary Actor
Receptionist

Secondary Actor
FAST social worker

Starting Point
The primary actor accesses the desk top system. Enters the information, prints and faxes the information to others.

Ending Point
The social worker receives the fax and decides on admission to the FAST program.

Information Exchanges
The receptionist provides first name, last name, address, telephone number, demographics,

drug reports, probation and arrest history.

Measurable Results
A new patient record is created and the clinician is notified of the event.

Figure 4: Alternate Documentation of a use case

Specify Fields

At the end of specifying system requirements you should have a list of fields that correspond to various uses of the database. The more details are specified in a Use Case the easier it is to extract the type of information that the system will require, and, consequently, the easier it will be to generate the database fields. The use cases gives us the entry point into the actual process of developing the information model for the target database. It helps us specify specific fields needed.

In assigning field names Hernandez (pages 209-211) suggests you follow these rules:

1. "Create a unique, descriptive name that is meaningful to the entire organization.
2. Create a name that accurately, clearly and unambiguously identifies the characteristics a field represents.
3. Use the minimum number of words necessary to convey the meaning of the characteristics the field represents.
4. Do not use acronyms and use abbreviations judiciously.
5. Do not use words that could confuse the meaning of the field name (e.g. the qualification digital is not needed in the field name digital identification code.)
6. Do not use names that implicitly or explicitly identify more than one characteristic (e.g. Phone/fax).
7. Use the singular form of the name.

We also add the following additional criteria:

1. Do not define fields that have multiple parts such as an a field that include the entire address and it does not separate the composite information into its components like street name, street number, zip code and so on. Combining all the information into one field will prevent future analysis and identification of individual components, e.g. we will not be able to know which other patient lives in the same zip code.
2. Each field should represent one single fact. Do not combine multiple facts into a field. For example, do not create a field for a value computed from several components. The database should keep the components and allow the value to be calculated within the software application.

End the process with a thorough and careful documentation. For each of the fields, provide the name of the field, a short definition, any restriction that pertains to the values the field and the type of values expected. Types of data allowed depend on the manufacturer of the database software you are using. Some common types are:

- Text field. Allows entry of alphabets as well as numbers but does not allow addition or multiplication of entry. Information in these fields can be appended to each other but not added.
- Binary fields. Allows entry of two mutually exclusive and exhaustive values such as Yes and No.
- Numeric fields. Allows entry of numbers that can be added or multiplied.
- Date fields. Allows entry of a month, day of the month and a year. Functions in the database can convert date fields to numbers and consequently calculate difference among dates.

Careful documentation and frequent sharing of the documentation with organizational members can help the design of databases.

In this lecture, we have seen how information requirements can be established from review of decisions and scenarios connecting actors to specific decisions. Use cases and scenarios begin to give us a handle on the kind of information we need to concentrate on. Within the use cases, we have learned that information flows are of great importance because they determine the fields to be kept in the database.