APPENDIX D
INTRODUCTION TO NIAM

NIAM (Nijssen's Information Analysis Methodology) was developed by Prof. G. M. Nijssen (University of Queensland) when he was working at Control Data Corporation. However significant contributions were made by E. D. Falkenberg (Katholieke Universiteit, Nijmegen). Over the years, NIAM has evolved owing to the research contributions of Nijssen, Falkenberg and others. In particular, several revisions and extensions to the methodology were developed by T. A. Halpin (University of Queensland).

Because of its emphasis on fact types, NIAM is also called fact-oriented modeling. Although similar in some respects to the Entity-Relationship model proposed by Chen (1976), NIAM provides a simpler and more natural approach to semantic modeling. Moreover, NIAM provides a high level procedure for relational database design that has many advantages over the traditional "normalization" technique. NIAM works on a Conceptual Schema Design Procedure (CSDP). In order to divide a complex task into manageable procedures, the CSDP is presented as a sequence of nine steps. These steps are as follows:

1. Transform familiar information examples into elementary facts, and apply quality checks.
2. Draw a first draft of the conceptual schema diagram, and apply a population check.
3. Eliminate surplus entity types and common roles, and identify any derived fact types.
(4) Add uniqueness constraints for each fact type.

(5) Check that fact types are of the right arity.

(6) Add entity type, mandatory role, subtype and occurrence frequency constraints.

(7) Check that each entity can be identified.

(8) Add equality, exclusion, subset and other constraints.

(9) Check that the conceptual schema is consistent with the original examples, has no redundancy and is complete.

The CSDP procedure begins with the analysis of information to be output by, or input to, the information system. Basically, the first three steps are concerned with identifying the fact types (stored or derived). In the following steps the constraints are added to the stores fact types. Throughout the procedure checks are performed to ensure that no mistakes have been made. The following sections are now devoted towards explaining the several steps which are involved in NIAM as concisely as possible, yet retaining their significance and clarity.
CSDP STEP 1: FROM EXAMPLES TO ELEMENTARY FACTS

If a conceptual schema is being designed for an application that was previously handled either manually or by computer, information examples will be readily available in the form of reports, documents etc. Two of the most important types of examples are output reports and input forms. These may be tabular, graphical, template or textual form. An example of output report is given in the next page. It is important to note that in order to transform examples into facts, it is important that the information should be familiar. To help understand this we use the telephone heuristic. That is we imagine that we are on the phone and have to convey the information in simple sentences, to the person at the other end of the line. The facts are now presented as follows, but rather loosely

EE 101 meets at Room Mon 3 p.m.

EE 101 is held in Room CS 101

Student 1234567 is in EE 101.

Student 1234567 has name Tom.

Since this verbalization involves some interpretation on our part it is important that the kind of example is familiar to us and to another person who may be assisting us with step 1. Input forms are used for data capture, that is getting specific information into the database. Sometimes information not appearing in final output report is required as data to enable the actual results to be computed. For example input forms for tutorial allocation are shown in the next page. Taken individually the output report
# AN OUTPUT REPORT INDICATING SCHEDULE OF CLASSES

<table>
<thead>
<tr>
<th>CLASS</th>
<th>TIME</th>
<th>ROOM</th>
<th>STUD. #</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>EE101</td>
<td>MON.3 PM</td>
<td>CS 101</td>
<td>1234567</td>
<td>TOM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2345678</td>
<td>DAVE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3456789</td>
<td>DICK</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4567890</td>
<td>ALLEN</td>
</tr>
<tr>
<td>ME202</td>
<td>TUE.2 PM</td>
<td>ME 234</td>
<td>1234567</td>
<td>TOM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2345678</td>
<td>DAVE</td>
</tr>
</tbody>
</table>

# EE 101 TUTORIAL PREFERENCE FORM

Please complete the form below to assist in tutorial allocations. Tutorials are of one hour duration, and are available at these times:

Monday | Tuesday | Thursday |
--- | --- | --- |
10 a.m. | | |
11 a.m. | | |
12 noon | | |
2 p.m. | 2 p.m. | |
3 p.m. | | 3 p.m. |

Student #:
Student Name:
Tutorial Preference 1:
Tutorial Preference 2:
and the input form reveal only partially the kinds of information needed for the system. In combination they might be enough for us to arrive at the structure of the UoD (Universe of Discourse). In this case the pair of examples are said to be significant.

Assuming that we have a significant set of familiar examples, we translate these examples into elementary facts. An elementary fact is a simple assertion about the UoD. Elementary facts are assertions that particular objects play particular roles. The simplest kind of elementary fact asserts that a single object plays a given role. For example: Robert sleeps. Here we have one object (Robert) playing a role (sleeps). Typically an elementary fact asserts that certain objects participate in a relationship. For example: Mike teaches Robert. Here we have two objects (Mike and Robert) participating together in a relationship (an academic instance). Before proceeding further there are two terms which need to be defined: entity and label. An entity is the basic object or thing that we talk about, such as Robert. In natural language, there are two ways of referring to entities. One method uses proper names: here a label is used to denote a particular object (e.g., George Bush). The other method is to use a definite description (e.g., The president of America).

Once we have transformed the information examples into elementary facts we should check if the entities are well identified and can the facts be split into smaller ones without losing information. One way of checking that we have included all components in our elementary facts is to draw a fact type-instance table. The top section of this table indicates the fact type being considered, by showing the entity types, reference modes and roles. Thus the top portion of the table deals only with the
conceptual schema. The bottom section of the table contains the actual labels stored in
the database. When read in conjunction with the top section, the rows in the bottom
provide instances of elementary facts. For instance, let's consider the table shown next
page along with the fact type-instance diagrams. If we consider step 1 on the first row,
we might at first consider expressing this information in terms of the following sentence :

The person with surname 'Wirth' designed the language with name 'Pascal' in the year
1971 AD.

If this is correct, we have a ternary fact type with the predicate :

..designed..in..

But we immediately recall that an elementary fact must be simple or irreducible. It
cannot be split into two or more facts in the context of the UoD. But clearly this fact
may be split into the following two elementary facts:

The person with surname 'Wirth' designed the language with name 'Pascal'.

The language with name 'Pascal' was designed in the Year 1971 AD.

If we know these two facts then we also know the original ternary fact we started with.

So, the ternary can be split. This results in the two fact type-instance diagrams shown
on the next page.
<table>
<thead>
<tr>
<th>Designer</th>
<th>Language</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wirth</td>
<td>Pascal</td>
<td>1971</td>
</tr>
<tr>
<td>Kay</td>
<td>Smalltalk</td>
<td>1972</td>
</tr>
<tr>
<td>Wirth</td>
<td>Modula-2</td>
<td>1979</td>
</tr>
</tbody>
</table>

**Fact Type-Instance Diagrams:**

**Entity Types:**

<table>
<thead>
<tr>
<th>Person</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Surname</td>
<td>Name</td>
</tr>
</tbody>
</table>

**Reference Modes:**

<table>
<thead>
<tr>
<th>Role</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designed</td>
<td>Wirth</td>
</tr>
<tr>
<td></td>
<td>Kay</td>
</tr>
<tr>
<td></td>
<td>Wirth</td>
</tr>
</tbody>
</table>

**Labels:**

<table>
<thead>
<tr>
<th>Label</th>
<th>Language</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pascal</td>
<td>1971</td>
</tr>
<tr>
<td></td>
<td>Smalltalk</td>
<td>1972</td>
</tr>
<tr>
<td></td>
<td>Modula-2</td>
<td>1979</td>
</tr>
</tbody>
</table>
The major work in this step consists of drawing a diagram which shows all the fact types. To do this we need a convention for illustrating entity types, label types and roles. Let us consider the following table:

<table>
<thead>
<tr>
<th>Drives</th>
<th>Person</th>
<th>Car</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adams</td>
<td>235PZN</td>
<td></td>
</tr>
<tr>
<td>Jones</td>
<td>235PZN</td>
<td></td>
</tr>
<tr>
<td>Jones</td>
<td>108AAQ</td>
<td></td>
</tr>
</tbody>
</table>

Let us try to express the information in this report by the following three elementary facts, using "reg#" as an abbreviation for "registration number".

The Person with name 'Adams' drives the Car with reg# '235PZN'.

The Person with name 'Jones' drives the Car with reg# '235PZN'.

The Person with name 'Jones' drives the Car with reg# '108AAQ'.

The conceptual schema diagram is shown in the next page. Entities and labels are denoted by ellipses and are both examples of objects. On a conceptual schema diagram the roles played by the objects are explicitly shown by boxes. The name of the role is placed inside or beside the box. Note that an n-ary fact type involves 'n' boxes.

Diagrammatically, we represent an 'n-ary' fact type as a contiguous sequence of 'n' role boxes, each of which is connected to exactly one entity type.
A CONCEPTUAL SCHEMA DIAGRAM

PERSON

DRIVES

DRIVEN BY

CAR

HAS NAME

IS NAME OF

PERSON NAME

HAS REG.

IS REG. OF

REG. #
CSDP STEP 3: TRIM SCHEMA AND FIND DERIVED FACT TYPES

The first part of this step is to look around to see if we have introduced any unnecessary entity types in our diagram. The basic approach is to partition the UoD in such a way that each entity is grouped into exactly one entity type; roughly, entities are grouped into the same type if we want to record similar information about them. So, in selecting entity types we should ensure that they are mutually exclusive, that is, they have no instances in common. For the moment we consider that entity types should not overlap. So, if we find entity types which do overlap then we should combine them into a single entity type. One reason for suspecting that two entity types should be combined is if they both have the same unit-based reference mode.

As an illustration let us consider the table shown in the next page. Note that the three entity types Wholesale price, Retail price and Markup all have the same unit-based reference mode ($). Moreover, looking at the table we see that $50 appears as both a wholesale price and a markup: since this same entity is an instance of both entity types the entity types overlap and hence should be combined. If the table population is significant the set of retail prices does not overlap the set of markups; nevertheless, it is meaningful to compare retail prices and markups since they have the same unit of dimension ($). These considerations lead us to collapse the three entity types into one. So, the fig.(2) shows how a conceptual schema diagram is corrected by applying step 3.
<table>
<thead>
<tr>
<th>ARTICLE</th>
<th>WHOLESALE PRICE ($)</th>
<th>RETAIL PRICE ($)</th>
<th>MARKUP ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>50</td>
<td>75</td>
<td>25</td>
</tr>
<tr>
<td>A2</td>
<td>80</td>
<td>130</td>
<td>50</td>
</tr>
<tr>
<td>A3</td>
<td>50</td>
<td>70</td>
<td>20</td>
</tr>
<tr>
<td>A4</td>
<td>100</td>
<td>130</td>
<td>30</td>
</tr>
</tbody>
</table>
A Faulty Conceptual Schema

Result of Applying Step 3 to Above Schema
CSDP STEP 4: ADD UNIQUENESS CONSTRAINTS FOR EACH TYPE

Thus far our conceptual schema design procedure has focussed on specifying the elementary fact types, both stored and derived. The rest of the CSDP is concerned mostly with specifying constraints. Constraints may be either static or dynamic. We focus on uniqueness constraints which are static constraints. Static constraints are described as being "constraints on the fact types". More accurately, static constraints apply to every possible state of a database. In this section representation of uniqueness constraints is described that apply to a single fact type of arity 1. Unary fact types are the easiest. As an illustration, suppose as a part of fitness application we are interested in joggers. This can be handled with a unary fact type, as shown in the figure in the next page where a small population is included. From the conceptual viewpoint the population of this fact type is the set containing the following facts:

The Person with surname 'Adams' jogs.
The Person with surname 'Brown' jogs.
The Person with surname 'Collins' jogs.

Given that these facts are recorded in the database, what would happen if we tried the following update operation:

add: The Person with the surname 'Adams' jogs.

The database is a variable whose value or population at any state is a set of elementary facts. Since the fact that Adams jogs is already present in the database, if this fact was now added the population would remain unaltered. From this point of view there is no problem in accepting the update. However, from the internal point of view, when an
elementary fact is added to a database it is typically stored in a previously unallocated space. So accepting the update operation above would mean that the fact that Adams jogs is actually stored twice, in two physically separate locations. This is an example of redundancy. If redundancy occurs we need to view the database as a bag of facts rather than as a set of facts.

The most important reason to avoid redundancy in a database is to maintain the integrity of the database by simplifying the correct handling of update operations. Managing updates would become very difficult if we allowed redundancy in the database. The second reason to avoid redundancy is to save space. On the other hand redundancy can be useful in making the retrieval of information more efficient, and sometimes to make information systems work fast enough it may be necessary to allow certain kinds of controlled redundancy, particularly with derived fact types. However, for the purpose of conceptual schema design we require to have no redundancy in the knowledge base. Since each row of a conceptual fact table corresponds to one elementary fact, this means that no row can be repeated. In our example, this means that the fact table on the right is illegal, since the Adams row is repeated. Here for any particular state of the database, each person can be recorded as being a jogger at most once. We represent this uniqueness constraint by placing a double headed arrow above the fact table. This indicates that each entry in the column must be unique. The conceptual schema constraint also appears next to the role, as is shown by the double arrow next to the role.
A SCHEMA-BASE DIAGRAM FOR A UNARY FACT TYPE

IDENTICAL SETS BUT DIFFERENT BAGS

ARROWED BAR NOTATION FOR UNIQUENESS CONSTRAINTS

UNIQUENESS CONSTRAINTS ARE SHOWN NEXT TO THE ROLE(S)
CSDP STEP 5: CHECK THAT FACT TYPES ARE OF THE RIGHT ARITY

If thus far the entire analysis has been carefully done, then this step is redundant. However, it is quite possible that some fact types may be too long or too short, thus requiring further checking. By "too long" is meant that the arity of the fact type is higher than it should be. In this case we must split the compound fact type up into two or more simple fact types. By "too short" is meant that the arity of some fact type is too small, resulting in loss of information. In checking the arity of the fact types in our conceptual schema there are at least three methods that may be of use. These are listed below:

(1) Make use of common sense or background knowledge of the UoD to decide if information is lost by splitting a fact type or if the fact type can be expressed as a conjunction of smaller ones.

(2) Use the splittability rules about the shortest key.

(3) Provide a significant fact table for the fact type, split this by projection then recombine by natural join: if new instances appear then the fact type is unsplittable in this way. Testing should be continued until a split is found or all ways are exhausted.

The first method is really the best and least formal and it is discussed. Consider the output report shown in the next page. Using our background knowledge of the UoD, we interpret the first row as stating that the person Adams is enrolled in the degree BSc and studies the subject CS112, scoring a rating of 7 for that subject. The "and" strongly suggests that we have a conjunction of two facts here, which we can set out as:

Person (surname) 'Adams' seeks Degree (code) 'BSc'.
A schema diagram for the above table is shown (flattened version)
Person (surname) 'Adams' enrolled in Subject (code) 'CS112', scoring Rating (Nr) 7. The word "seeks" means "is enrolled to obtain the qualification". This kind of a fact type is best represented by a flat ternary as shown in the next page. Note that a ternary fact type is elementary.
CSDP STEP 6: ADD ENTITY TYPE, MANDATORY ROLE, SUBTYPE AND OCCURRENCE FREQUENCY CONSTRAINTS

Besides uniqueness constraints there are several other constraints that need to be considered. The most common of these are: entity type constraints, mandatory roles, subtype constraints and occurrence frequencies. This section is quite complex and is not elaborated. Interested readers are requested to refer the book by Nijssen.
CSDP STEP 7: CHECK THAT EACH ENTITY CAN BE IDENTIFIED

Usually a 1:1 naming convention is used to identify entities of a given type. Sometimes however, the identification scheme is more complex. In each case, we need to ensure that the means by which entities are references are clearly defined, both for the system and for the user. Checking this is the next step of the conceptual procedure. We consider as an illustration a simple 1:1 case as shown in the table next page. This output report indicates subjects studied by students. Each student has a single, distinct name and each subject has a single, distinct code. These 1:1 reference schemes are specified by naming the reference modes in parentheses. We might also specify lexical constraints as shown. The lexical constraint on student names is weak, for example, we have not excluded students name like "CS100". Entities in the real world are identified by singular terms in the knowledge base. Fundamentally, such singular terms are always definite descriptions. For example, a particular student is identified by the term "the student with name 'Anderson P'". However, within the context of a reference type, a label is said to be an identifier for an entity if the label denotes only one entity of that type.

While names and codes serve to identify students and subjects, humans tend to read more into these reference schemes than what the conceptual schema diagram suggests. Such as, when we see the name "Anderson P", we tend to assume that "Anderson" is a surname and "P" is the first initial of the student. So long as we don't require the system to possess this deeper insight into the naming convention, we can leave the schema as it is. If we wanted to list all students with the surname 'Jones', this schema cannot provide
CSDP 7: A CONCEPTUAL SCHEMA DIAGRAM FOR THE TABLE ABOVE.

A REFERENCE SCHEME REQUIRING TWO LABELS FOR TABLE ABOVE.
the required the answer since it doesn't know anything yet about how surnames fit into
the notion of student names. There are several ways in which the additional semantics
may be formalised. One way would be to split student names into two components as
shown in the same page. For this UoD, surnames are not identifiers, nor are initials.
However, the combination of the surname and initials does provide an identifier since
this combination refers to only one student, as indicated by the inter-reference type
uniqueness constraint.
CSDP STEP 8: FURTHER CONSTRAINTS

There are a variety of constraints that can be specified. These include uniqueness constraints, entity type constraints, mandatory roles, subtype constraints and occurrence frequencies. Such constraints hold for all states of the database, and are enforceable by the system. Although these cover the vast majority of constraints that are of practical significance, other kinds of constraint sometimes need to be applied. The specification of these further constraints is the next step in the conceptual schema diagram procedure.

We will discuss the equality constraint only in this section. Suppose a health club maintains an information system concerning its members, and that the table shown in the next page is an extract of an output report from this system. As usual, we use "?" for a null value, although in reports a blank is often used instead. Note that the reaction time and heart rate information is optional. If we look at the first and the second rows, we will find nothing to let the system know that reaction time and heart rate should be noted for Anderson and not Bloggs. Since the system has no well defined reason for deciding whether these roles should be recorded, we simply leave them as optional roles attached to a member rather than attempt subtyping.

Assuming that the report is significant in this respect, reaction time is recorded for a member if, and only if, the resting heart rate is. We can imagine a scenario in which this constraint makes sense. Both these figures are measures are fitness. If a member feels inclined to take one of these tests the health club arranges for that member to also
<table>
<thead>
<tr>
<th>MEMBER</th>
<th>SEX</th>
<th>BIRTH YR</th>
<th>SPORT</th>
<th>REACTION TIME</th>
<th>RESTING HEART</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANDERSON</td>
<td>M</td>
<td>1940</td>
<td>GOLF</td>
<td>250</td>
<td>80</td>
</tr>
<tr>
<td>BLOGGS</td>
<td>M</td>
<td>1940</td>
<td>GOLF</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>FIT I.M.</td>
<td>F</td>
<td>1960</td>
<td>Aerobics Tennis</td>
<td>250</td>
<td>80</td>
</tr>
<tr>
<td>HUME P.E.</td>
<td>F</td>
<td>1946</td>
<td>GOLF</td>
<td>305</td>
<td>93</td>
</tr>
</tbody>
</table>

**CSDP 8: A CONCEPTUAL SCHEMA DIAGRAM FOR THE ABOVE TABLE**

r1 = has
r2 = has reaction time
r3 = was born in
r4 = has resting heart rate
r5 = plays
take the other test so as to ensure a balanced view of fitness rather an incomplete one. When the data are entered, a compound transaction is used to enter both figures. This constraint is expressed diagramatically by means of a double headed arrow between the relevant roles as shown in the conceptual schema diagram. Such a constraint is called an equality constraint, since for any state of database the set of people for which \( r2 \) is recorded equals the set of people for which \( r4 \) is recorded.
The conceptual design procedure facilitates early detection of errors by various checking arrangements including ongoing feedback to the user by way of examples. Nevertheless, it is wise to include a final check to pick up any errors that might have slipped through. This step is the last step in the conceptual schema design procedure. An important check in this stage is the population check. As an illustration consider the table and the schema diagrams shown in the next page. Notice the null value on row 3. We know that Bob seeks a BA degree but we do not know any of Bob’s subjects. We arrive at the schema shown below the table. To check on the constraints, let's populate the fact tables with our sample data. This gives the next schema diagram. We consider first the uniqueness constraint on the Studies fact type. This asserts that each entry in the person column is unique. However, the entry Ann appears twice here. So, this constraint is wrong. The population of the fact table makes it clear that the uniqueness constraint should span both columns.

Now consider the uniqueness constraint on Person seeks degree. This asserts that each entry in the degree column is unique. But the entry BSc appears twice there, and so does the entry BA. The presence of either of these cases proves that this constraint is wrong. The corrected conceptual schema is shown in the following page.
CSDP 9

<table>
<thead>
<tr>
<th>PERSON</th>
<th>SUBJECT</th>
<th>DEGREE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANN</td>
<td>CS112</td>
<td>BSc</td>
</tr>
<tr>
<td>ANN</td>
<td>CS100</td>
<td>BSc</td>
</tr>
<tr>
<td>BOB</td>
<td>?</td>
<td>BA</td>
</tr>
<tr>
<td>SUE</td>
<td>CS112</td>
<td>BA</td>
</tr>
</tbody>
</table>

IS THIS A CORRECT CONCEPTUAL SCHEMA FOR ABOVE TABLE?

THE CONSTRAINTS ARE INCONSISTENT WITH THE POPULATION
A correct conceptual schema for the table for CSDP 9.