Based on handout:

Adaptable methodology for database design
by N. Roussopoulos and R.T. Yeh, IEEE
Computer Vol. 17, no. 5, pp. 64-80. 1984

Goal

- Given an English description of an enterprise
- build a system to automate it and
- produce the documentation

In diagram form
- tasks
- documents
• Students register
• Students enroll in courses
• Students ask for transcripts
• Students enroll in courses
• Students register

Running example - 'Mini-U'

Phase-I

Phase-II

1. Read Account
   - Account
   - Read Account
2. Read Courses
   - Courses
   - Read Courses
3. Read Grades
   - Grades
   - Read Grades
4. Read Transcripts
   - Transcripts
   - Read Transcripts
Requirement analysis

Turn English description into top level information flow diagram, where
- boxes -> documents (~ db tables)
- ovals -> tasks (= db programs)

Important: system boundary

Top level info. flow diagram

System boundary

- internal documents -> db tables
- tasks -> db programs
- tasks: internal only
More on top level diagram

- reg. form
- enroll. form
- reg.
- enroll
- student rec.
- class rec.

More on top level diagram

- reg. form
- enroll. form
- transc. req.
- transcr.
- reg.
- enroll
- transcr.
- student rec.
- class rec.

Example - Mini-U

- Students register
- Students enroll in courses
- Students ask for transcripts
- Administrator records grades
- every semester: print class rosters
Document + Task forms

Top level diagram: only half of the info - we also need:
• Document forms and document list
• Task forms and task list

Document list

• D1: registration form
• D2: enrollment for
• …
• D7: student record
• D8: class record

} INTERNAL

Document forms

• D1: registration
  – ssn
  – name
  – address

• D2: enrollment
  ssn
  name
  List-of:
    course id
    course name
Document forms - cont’d

D3: transcript request form
- ssn
- name

D4: transcript
- ssn
- name
- List-of:
  - class-id
  - class name
  - grade

Document forms - cont’d

D7: student record
- ssn
- name
- address

Document forms - cont’d

D8: class record
- class-id
- class-name
- syllabus
- List-of
  - ssn
  - grade
Document forms - cont’d

• IMPORTANT POINTS
  – avoid redundancy in internal documents: i.e.,
    grades should be stored in ONE place only
  – there are many, different, correct solutions

Task List

• T1: Registration
• T2: Enrollment
• T3: Transcript
• ...

Task forms

• As in [R+Y]
• not required for this homework
• sub-tasks: probably there won’t be any
  – otherwise: ~3-7 sub-tasks per task
Database schema - E-R

- from the internal documents
- use their forms
  - ‘List-of’ constructs -> relationships

Eg., for ‘Mini-U’:
  D7: Student record (ssn, name, address)
  D8: Class record (c-id, …, List-of … )

E-R diagram for Mini-U
Relational schema

student (ssn, name, address)
class (c-id, c-name, syllabus)
takes (c-id, ssn, grade)

Make sure that
  – Primary keys are underlined;
  – tables are in BCNF (or 3NF at worst)

SQL DDL statements

create table student (ssn char(9), …);
create table class (c-id char(5), …);
...

Phase-I

description
  (req. anal)
top level I.F.D
  (sys. anal)
task + doc forms

conc. mod.
schema.
task email
pseudo-code

Phase-II

impl. + test.
code.
tests
user’s man.
### Task emulation

**T1: Registration**

read ssn, name and address

if ( ssn does not exist in 'student'){
   insert into student values ( :ssn, :name, :address);
} else{print “error: duplicate ssn”}

---

### Testing

- For T1 (registration), we check
  - duplicate ssn
  - ssn with 9 digits
- For T2 (enrollment), we check
  - for valid ssn (9 digits)
  - for registered ssn
  - for valid c-id
  - for duplicate (ssn, c-id) entry
User’s manual

Short (~1 page or less) - eg.,:
• copy myproject.tar
• do ‘make’
• follow the menu
<anything else the user should know, like OS, space requirements, etc etc>

Important points for Phase-I

• No redundancy in the fields of internal documents
• don’t forget the system boundary
• make sure the top level diagram agrees with the internal document forms
• explain if/when we deviate from BCNF