Carnegie Mellon Univ.
Dept. of Computer Science
15-415/615 - DB Applications

Lecture #19 (not in book)
Database Design Methodology handout

Based on handout:

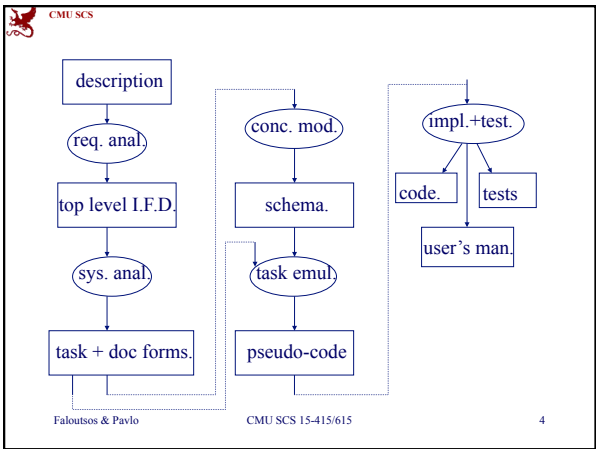
Adaptable methodology for database design
by N. Roussopoulos and R.T. Yeh, IEEE Computer Vol. 17, no. 5, pp. 64-80. 1984

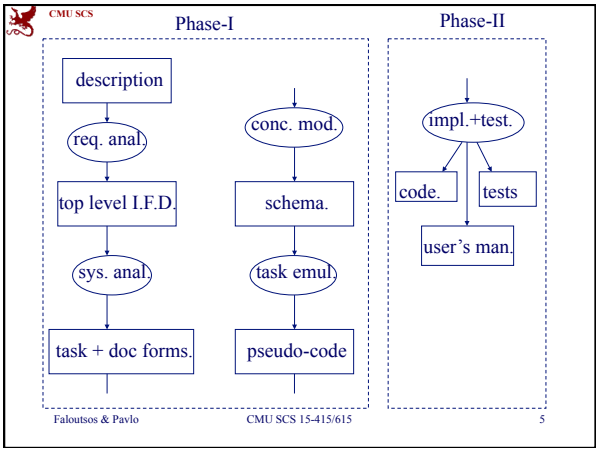
Goal

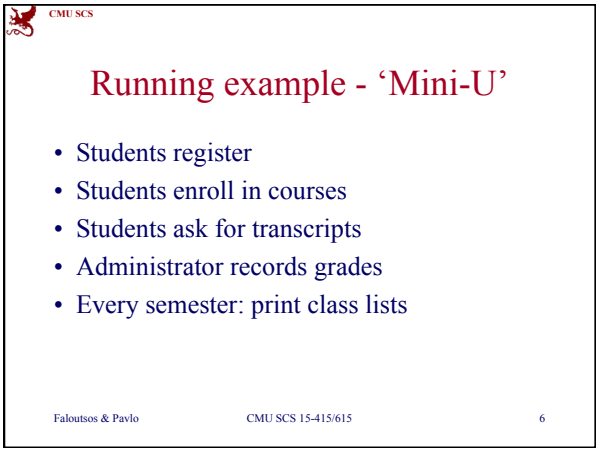
- Given an English description of an enterprise
- build a system to automate it and
- produce the documentation

In diagram form

- tasks 
- documents 


CMU SCS
description
req. anal.
top level I.F.D.
sys. anal.
task + doc forms.
conc. mod.
schema.
task emul.
pseudo-code
impl.+test.
code.
tests
user's man.
Faloutsos & Pavlo
CMU SCS 15-415/615
4


CMU SCS
Phase-I
Phase-II
description
req. anal.
top level I.F.D.
sys. anal.
task + doc forms.
conc. mod.
schema.
task emul.
pseudo-code
impl.+test.
code.
tests
user's man.
Faloutsos & Pavlo
CMU SCS 15-415/615
5


CMU SCS
Running example - 'Mini-U'
• Students register
• Students enroll in courses
• Students ask for transcripts
• Administrator records grades
• Every semester: print class lists
Faloutsos & Pavlo
CMU SCS 15-415/615
6

CMU SCS

Requirement analysis

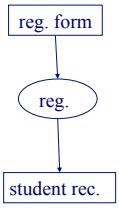
Turn English description in to **top level information flow diagram**, where

- boxes -> documents (~ db tables)
- ovals -> tasks (= db programs)

Important: **system boundary**

CMU SCS

Top level inf. flow diagram

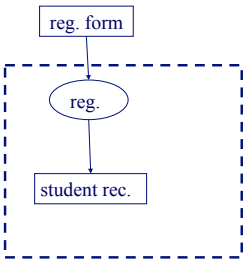


input

output

CMU SCS

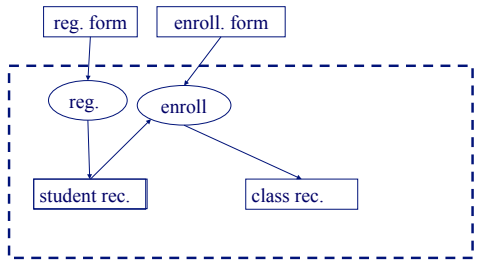
System boundary



- internal documents -> db tables
- tasks -> db programs
- tasks: internal only

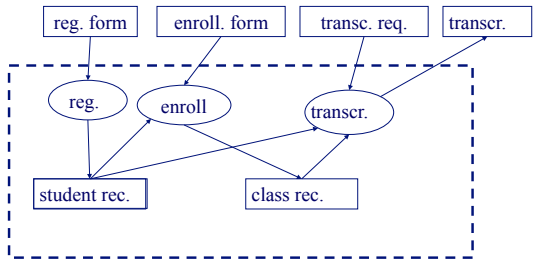
CMU SCS

More on top level diagram



CMU SCS

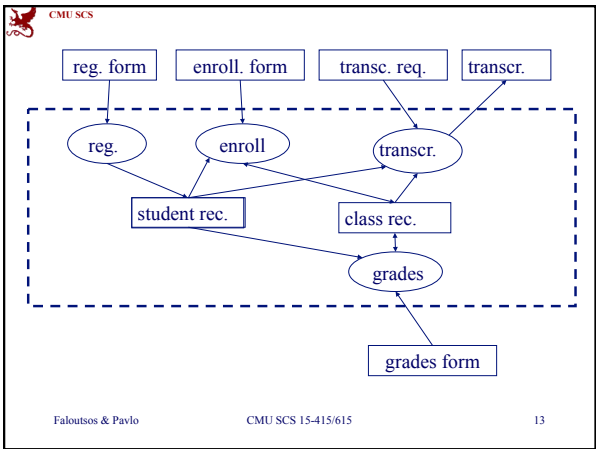
More on top level diagram

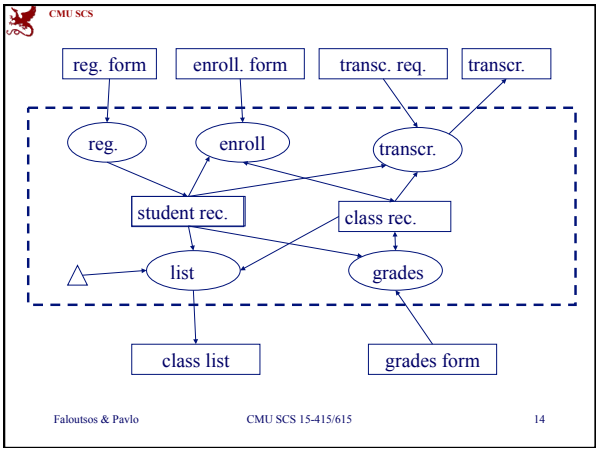


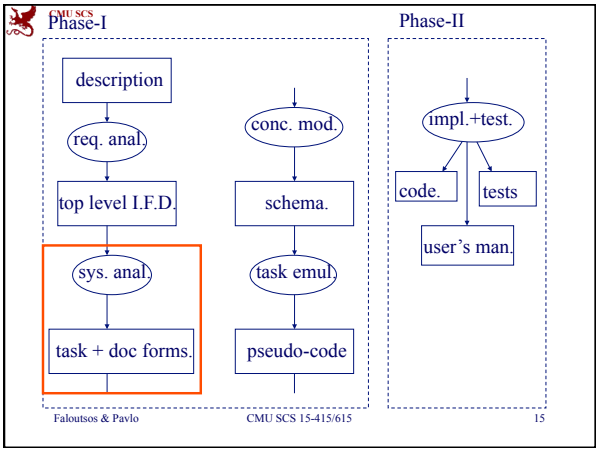
CMU SCS

Example - Mini-U

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


CMU SCS
reg. form
enroll. form
transc. req.
transcr.
reg.
enroll
transcr.
student rec.
class rec.
grades
grades form
Faloutsos & Pavlo
CMU SCS 15-415/615
13


CMU SCS
reg. form
enroll. form
transc. req.
transcr.
reg.
enroll
transcr.
student rec.
class rec.
list
grades
class list
grades form
Faloutsos & Pavlo
CMU SCS 15-415/615
14


CMU SCS
Phase-I
Phase-II
description
req. anal.
top level I.F.D.
sys. anal.
task + doc forms.
conc. mod.
schema.
task emul.
pseudo-code
impl.+test.
code.
tests
user's man.
Faloutsos & Pavlo
CMU SCS 15-415/615
15

CMU SCS

Document + Task forms

Top level diagram: only half of the info - we also need:

- Document forms and document list
- Task forms and task list

CMU SCS

Document list

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

} INTERNAL

CMU SCS

Document forms

- D1: registration
  - ssn
  - name
  - address

D2: enrollment
ssn
name
List-of:
course id
course name

CMU SCS

Document forms - cont'd

- D3: transcript request form
  - ssn
  - name

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

CMU SCS

Document forms - cont'd

(Internal documents - VERY IMPORTANT)

D7: student record
- ssn
- name
- address

CMU SCS

Document forms - cont'd

D8: class record
- class-id
- class-name
- syllabus
- List-of
  - ssn
  - grade

CMU SCS

Document forms - cont'd

- IMPORTANT POINTS
  - avoid redundancy in internal documents: ie., grades should be stored in ONE place only
  - there are many, different, correct solutions

CMU SCS

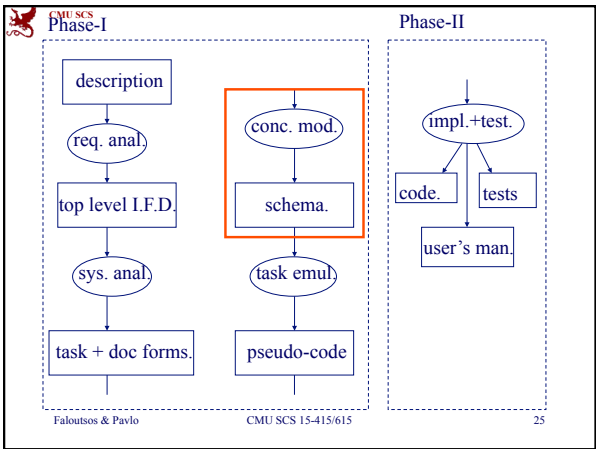
Task List

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

CMU SCS

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



CMU SCS

Database schema - E-R

- from the **internal** documents
- use their forms
  - 'List-of' constructs -> relationships

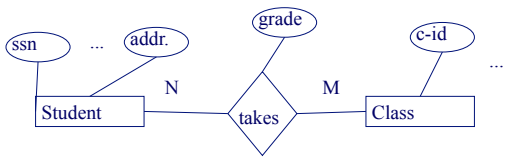
Eg., for 'Mini-U':

D7: Student record (ssn, name, address)

D8: Class record (c-id, …, List-of …)

CMU SCS

E-R diagram for Mini-U



CMU SCS

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)

CMU SCS

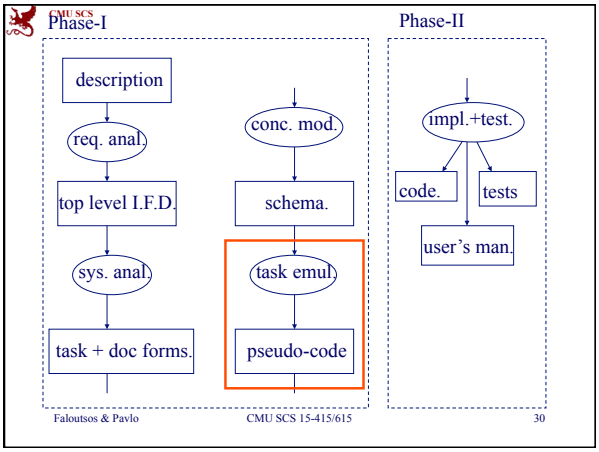
SQL DDL statements

create table student (ssn char(9), …);

create table class (c-id char(5), …);

…

CMU SCS



CMU SCS

Task emulation

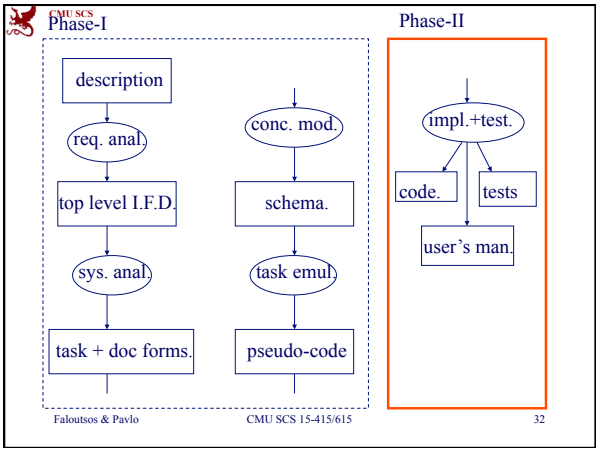
T1: Registration

read ssn, name and address

if (ssn does not exist in 'student'){

insert into student values (:ssn, :name, :address);

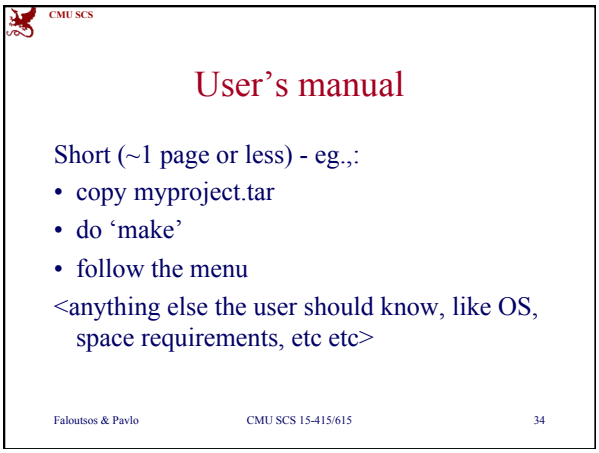
} else{print "error: duplicate ssn"}

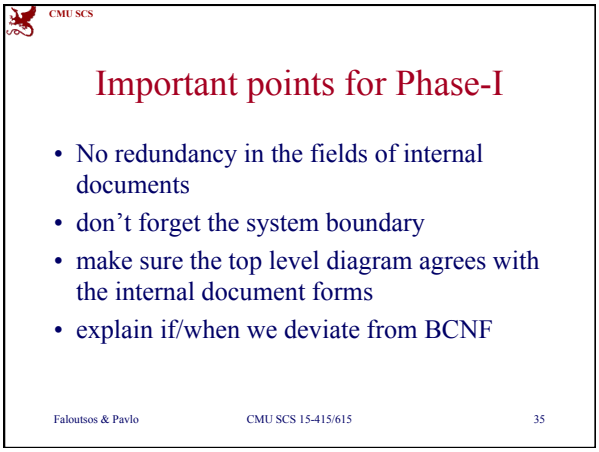


CMU SCS

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


CMU SCS
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>
Faloutsos & Pavlo CMU SCS 15-415/615 34


CMU SCS
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
Faloutsos & Pavlo CMU SCS 15-415/615 35