BLG411E - Software Engineering
Midterm Exam - 11.11.2008 (KEYS)

Q1 a) [15 points]

External Inputs (EI): (Total=5)

- Getting a customer's name for a transaction (service request, inquiry)
- Customer details screen
- Service request details screen
- Service operation details screen
- Menu selection

External Outputs (EO): (Total=9)

- List of requests by service registration number.
- List of requests by customer name.
- List of requests by status code.
- List of requests by device type.
- List of requests by request type.
- List of requests by date of request.
- List of requests by technician name.
- Billing document.
- List of devices sorted by warranty expiration date.

External Inquiries (EQ): (Total=1)

- Output screen of query to see the Status Codes of a given customer's all service requests.

Internal Logical Files (ILF): (Total=3)

- Customers table
- Service Requests table
- Technicians table

Unadjusted FP

Type of Component	Count	Average Weight	Total
External Inputs (EI)	5	x 4	20
External Outputs (EO)	9	x 5	45
External Inquiries (EQ)	1	x 4	4
Internal Logical Files (ILF)	3	x 10	30
External Interface Files (EIF)	0	x 7	0

TOTAL= 99

LOC = 99 UFP * 70 LOC/FP

= 6930 lines of code (in PHP language)

KLOC ≈ 7

Q1 b) [15 points]

COCOMO II Early Design Model Effort Multipliers

	Cost Driver	Our Estimate
1	PERS (Personnel capability)	High (0.83)
2	RCPX (Product reliability and complexity)	Nominal (1.00)
3	RUSE (The reuse required)	Low (0.95)
4	PDIF (Platform difficulty)	Low (0.87)
5	PREX (Personnel experience)	Very Low (1.33)
6	FCIL (The team support facilities)	Low (1.10)
7	SCED (Required schedule)	Nominal (1.00)

$$\prod_{j=1}^{7} \mathrm{EM}_j = 1.004$$

COCOMO II Scale Factors

	Scale Factors	Our Estimate
1	**PREC** (Precedentedness)	Nominal - somewhat unprecedented (3.72)
2	**FLEX (**Development Flexibility)	High - general conformity (2.03)
3	**RESL** (Architecture/Risk Resolution)	Nominal - often (60%) (4.24)
4	**TEAM** (Team Cohesion)	Very High - Highly cooperative (1.10)
5	**PMAT** (Process Maturity)	Low - CMM Level 1 (upper half) (6.24)

$$\sum_{j=1}^{5} \mathrm{SF}_j = 17.33$$

$$E = B + 0.01 * \sum_{j=1}^{5} \mathrm{SF}_j = 0.91 + 0.01 * 17.33 = 1.0833\ (\mathit{Exponent}\ )$$

$$F = D + 0.2 * (E - B) = 0.28 + 0.2 * (1.0833 - 0.91) = 0.3147\,(\mathit{Exponent})$$

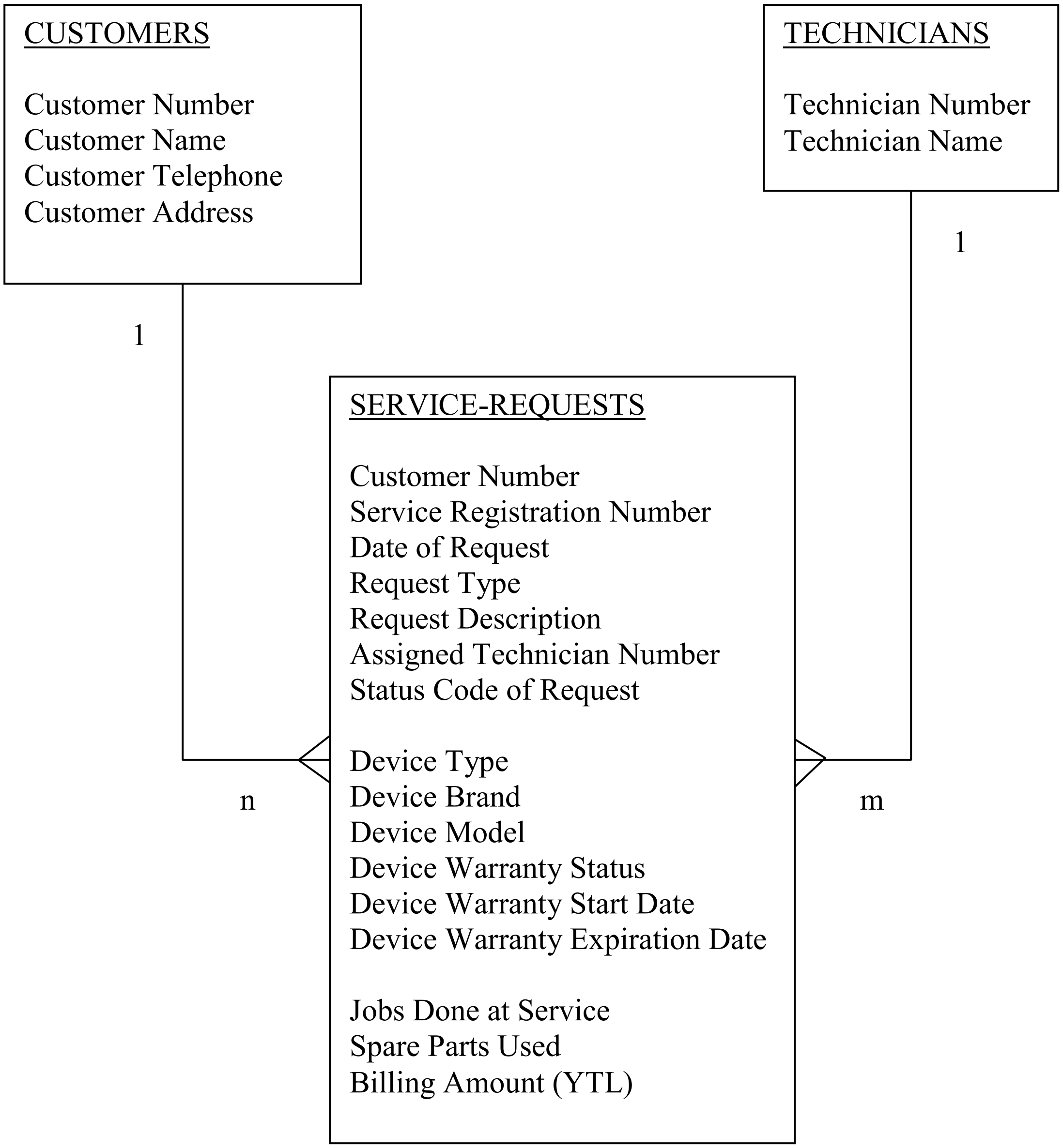
$$\mathrm{PM} = A * (\mathrm{KLOC})^{E} * \prod_{j=1}^{7} \mathrm{EM}_j = 2.94 * (7)^{1.0833} * 1.004 \cong 24\ (\text{Effort in Person - months})$$

$$\mathrm{TDEV} = C * (\mathrm{PM})^{F} = 3.67 * (24)^{0.3147} \cong 10\ (\text{Development Time in Months})$$

Number of people = PM / TDEV = 24 /10 ≈ 3

Q1 c) [15 points]

Entity-Relationship Diagram (ERD)



CUSTOMERS

Customer Number
Customer Name
Customer Telephone
Customer Address

TECHNICIANS

Technician Number
Technician Name

SERVICE-REQUESTS

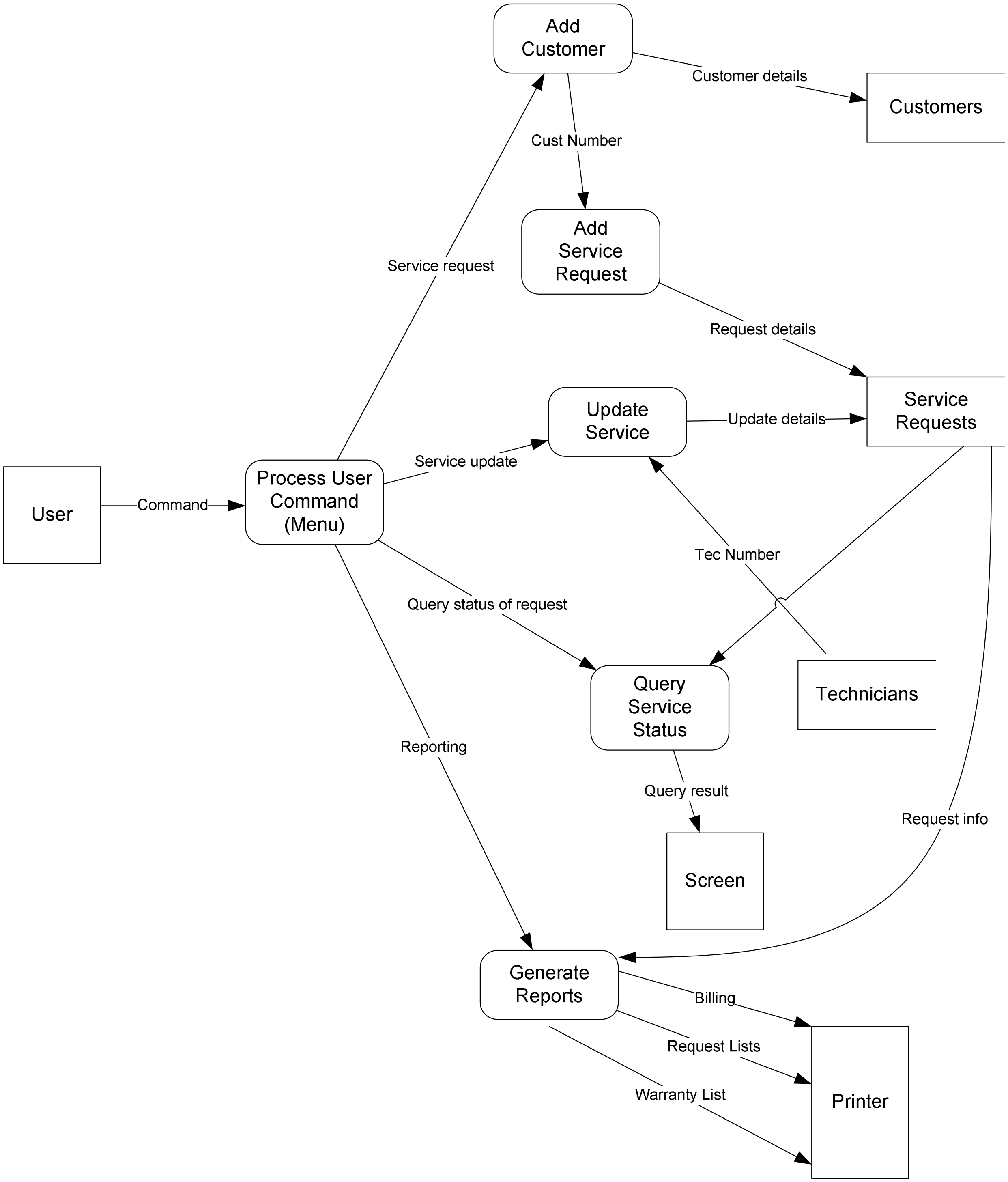
Customer Number
Service Registration Number
Date of Request
Request Type
Request Description
Assigned Technician Number
Status Code of Request

Device Type
Device Brand
Device Model
Device Warranty Status
Device Warranty Start Date
Device Warranty Expiration Date

Jobs Done at Service
Spare Parts Used
Billing Amount (YTL)

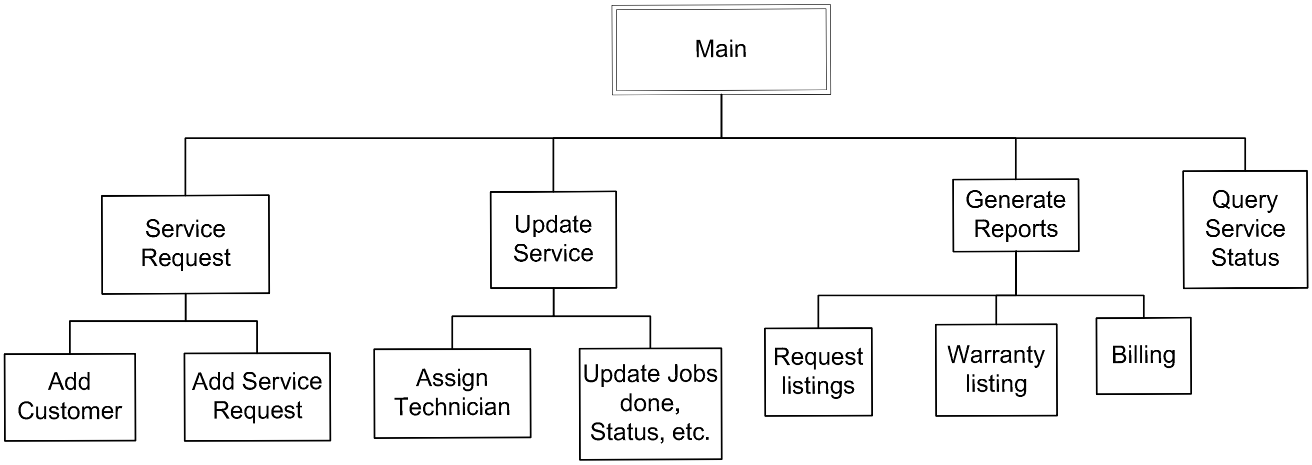
Q1 d) [15 points]

Level-1 Data Flow Diagram (Gane-Sarson notation)



Q1 e) [10 points]

Hierarchical Structure Chart



Q2) [15 points]

COCOMO II Sub-models

Early design model: Used when requirements are available but design has not yet started.
Post-architecture model: Used once the system architecture has been designed and more information about the system is available.
Reuse model: Used to compute the effort of integrating reusable components.
Application composition model: Used when software is composed from existing parts.

SUB MODEL	**BASED ON**	**USED FOR**
Early design model	Number of function points	Initial effort estimation based on requirements
Post-architecture model	Number of lines of source code	Development effort based on design specification
Reuse model	Number of lines of code reused or generated	Effort to integrate reusable components or automatically generated codes
Application composition model	Number of application points	Prototyping with scripts, DB programming etc.

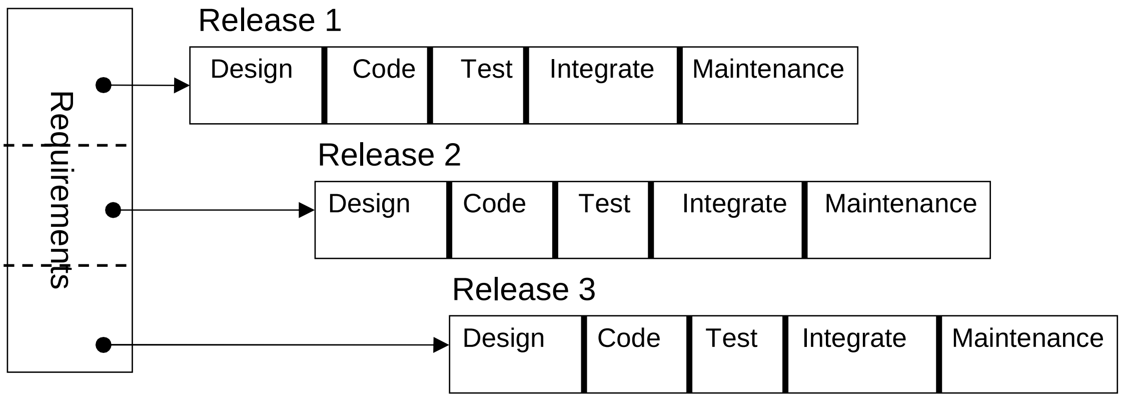
Q3) [15 points]

Incremental Software Process Model

- Avoids “big bang” implementation
- Assumes all requirements known up-front
- Each release adds more functionality
- The development and delivery is broken down into increments with each increment delivering part of the required functionality.
- User requirements are prioritised and the highest priority requirements are included in early increments.
- Once the development of an increment is started, the requirements are frozen though requirements for later increments can continue to evolve.

Advantages:

- Increments act as a prototype to help elicit requirements for later increments.
- The highest priority system services tend to receive the most testing.
- Good for projects with intensive user interfaces.
- Preferred when project deadlines are tight.


Release 1
Design
Code
Test
Integrate
Maintenance
Requirements
Release 2
Design
Code
Test
Integrate
Maintenance
Release 3
Design
Code
Test
Integrate
Maintenance