SEP REVIEW

Contents

6 Formal Specification

Name

Negation
Conjunction
Disjunction
Implication
Equivalence

Syntax

::= Predicate

Predicate

Predicate Predicate

| Predicate Predicate
| Predicate Predicate
| Predicate Predicate

These connectives are shown in decreasing order of binding power; the connective associates to the right, and the other binary ones associate to the left.

Description

These are the connectives of propositional logic; they allow complex predicates to be built from simpler ones in such a way that the truth or falsity of the compound in some binding depends only on the truth or falsity of the parts in the same binding. For example, the predicate P1 P2 is true in any binding if and only if both P1 and P2 are true in that binding. The following table lists the circumstances under which each kind of compound predicate is true:

P P is not true
P1 P 2 Both P1 and P2 are true
P1 P 2 Either P1 or P2 or both are true
P1 P 2 Either P1 is not true or P2 is true or both
P1 P 2 Either P1 and P2 are both true, or they are both false

Name

Schema	negation
Schema	conjunction
Schema	disjunction
Schema	implication
Schema	equivalence

Syntax

::= Schema-Exp

Schema-Exp

Schema-Exp Schema-Exp

Schema-Exp	Schema-Exp
Schema-Exp	Schema-Exp
Schema-Exp	Schema-Exp

Description

These are the logical operations on schemas which were introduced in Section 2.2.3. The negation S of a schema S has the same signature as S , but its property is true in just those bindings where the property of S is not true. For one of the binary operations to be allowed, its two arguments must have type compatible signatures. The signatures are joined to form the signature of the result. The truth of its property in any binding z is defined in terms of the truth in the argument schemas of the restrictions of z to their signatures. For example, the property of S T is true in a binding z if and only if either the property of S is true in the restriction of z to the signature of S , or the property of T is true in the restriction of z to the signature of T (or both). The other operations follow the rules for propositional connectives (see page 69).

THE UNIVERSITY OF ADELAIDE
AUSTRALIA

Examination for
Bachelor of Agricultural Science, Bachelor of Architectural Studies, Bachelor of Computer Science, Bachelor of Mathematical and Computer Science, Bachelor of Business Information Technology, Bachelor of Engineering, and the Graduate Diploma in Computer Science

Semester 2, November 2007

3675, 6263 Software Engineering and Project COMPSCI 3006, 7015

Official Reading Time:	10 mins
Writing Time:	120 mins
Total Duration:	130 mins

Questions	Time	Marks
Answer all 7 questions	120 mins	120 marks 120 Total

Instructions

- Begin each answer on a new page
- Examination material must not be removed from the examination room
- Simple, Non-programmable Calculators Allowed
- Text book, Lecture notes, Personal notes, Foreign language dictionary (paper), English language dictionary (paper) Allowed

Materials

- 1 Blue book

DO NOT COMMENCE WRITING UNTIL INSTRUCTED TO DO SO

Task Management

Question 3

If an incremental development model has been chosen (as in Question 1) we need to create a project time line that reflects this chosen process.

(a) Identify a series of milestones (at least 4) for the new project.

[8 marks]

(b) Draw a project timeline for the new full version robot system, assuming that the starting point for the project is the city mapping robot system already developed by your group. The project timeline should include the milestones identified above and should show the important aspects of the incremental development model.

Note that you do not need to provide any time estimates for any work unit.

[8 marks]

[Total for Question 3: 16 marks]

1. GUI Mockups

2.

① project analysis and gathering

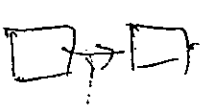
② initial planning

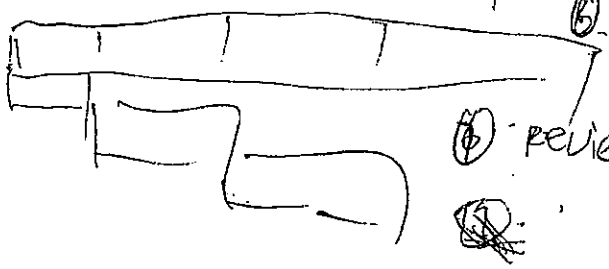
③ Design

④ coding

⑤ feedback from client

⑥ testing





⑦ review

⑧

Risks

Question 4

Initial estimates indicate that it will take your group 12 months, working full-time, to develop the full version of the city mapper software. You and your fellow group members are all keen to at least start on the project, but there is already some doubt that you will all be available for the full 12 months. Indeed the best programmers in your group have already applied for other, longer term positions, with other companies. With this in mind, there is a real risk of staff turnover in this project.

(a) Write an appropriate entry, one that might appear in the SPMP, for this risk. The entry should cover the following:

- Risk description.
- Risk likelihood. (Justify your answer.)
- Risk severity. (Justify your answer.)
- Actions in place to minimise likelihood of risk.
- Actions in place to minimise severity of risk.
- Actions to be taken if risk occurs.

[12 marks]

(b) Based upon your experience in the robot project in this subject suggest two problems that might occur in implementing the above risk management plan.

[4 marks]

[Total for Question 4: 16 marks]

Likelihood: very low, low, moderate, high or very high

severity: Catastrophic, serious, tolerable, insignificant.

Quality Management

Question 7

(a) Your team will be expected to continue generating a large amount of documentation (including SRS, SPMP and SDD documents). Describe a process that you would put in place to write and review documentation before it is released. Compare this process with what you did in your project (be honest), and describe how your new process is an improvement on the old process. draft → draft review → final → fn

[8 marks]

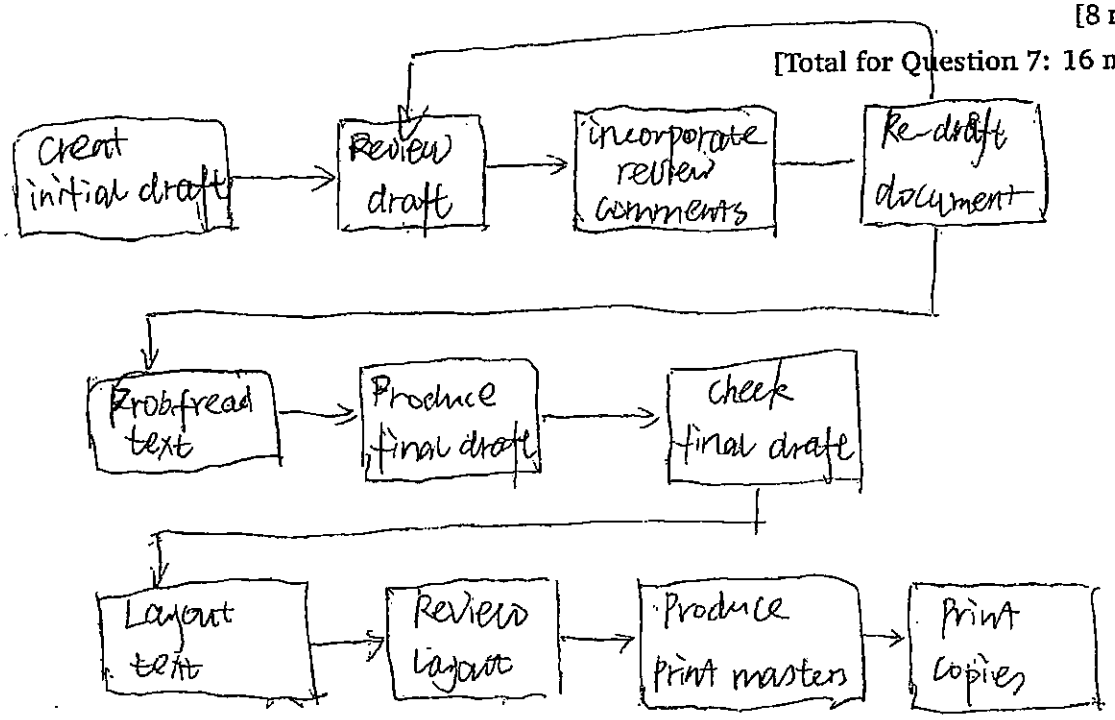
(b) Briefly describe 2 processes or methods that you would employ to develop code of a high quality. For each process or method, explain why it can improve the overall quality.

print

LayoutEdit

[8 marks]

[Total for Question 7: 16 marks]



End of exam

Sept (2008) Date: No

Q1.

① give a description about engineering process (which can be found in spp)

b). the models — give details — what is for each part and give dates of it.

c). give four reasons: the identification of risk is incorrect. risk. change the model mystery is not bad.

people model: ① give feedback

② easy for interupts change

Q2.

a). IDE. Integrated Development Environment

b). Makefile & Antfile (be able to write it)

Q3:

(1). how to build the project; demonstrate it to the user

① set up an interface for information (read senson.)

② read GPS so that you know the location / get the information from the GPS

③ Individual information you wanna say?

b). What tasks we need to do to complete milestones?

① write code to read GPS sensors ② testing ③ requirements for garuntee ④ Initial planning

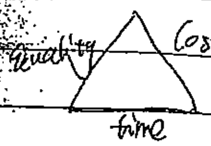
summrize the task list.

Don't give the time task, indicate the orders.

Date: No.

Q4.

a) ~~Reading~~ and understanding risks.



safety risk certification
: put more testing — minimise the risk likelihood

~~Minimise~~ severity of risk : { good interface / good design
| suitability

Actions taken if risks occur.

① ~~get~~ early warnings, negotiate with the clients

Q5.

a). Week 4's lecture.

Break down whole system into pieces of ~~sy~~ subsystems.

how to organize ? — ~~Repossery~~ Repository model / client-server / layered.

? Brown case model ...

~~FP~~ { object-oriented | — decompose the subsystems.
{ functional pipeline |

control style — how to communicate with each other.

① centralized controller. pick up a particular one to control other

② ~~eternate~~ events ...

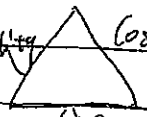
③ Interrupt — reserve the memory

c). advantages & limitations.

Date: No

Q4

a) Testing and understanding risks.



safety risk certification
: put more testing - minimise the risk likelihood.
minimise severity of risk: good interface / good design
suitability

Actions taken if risks occur.

① ~~set~~ early warnings, negotiate with the clients.

Q5

a) week 4's lecture.

break down whole system into pieces of ~~sf~~ subsystems.

how to organize: ~~Repository~~ Platform
~~Repository~~ model / client-sever layered.
= Brown coast model --

It: { object-oriented
functional pipeline } decompose the subsystems.

control style — how to communicate with each other.

① centralized controller - pick up a particular one to control other

② ~~event~~ Events ---

③ Interupts — reserve the memory

c) advantages & limitations.

Date: No

Q6.

a). think about gps

use cases

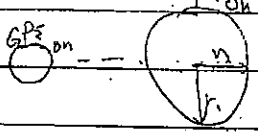
b). ~~functional user requirement~~

Title ① turn gps off for 5 minutes. then turn it on. it comes up with these words

Description

Rationale

Acceptance Criteria



② make the robot ~~constraints~~ online.

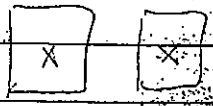
how the robot moves.

how many steps does the robot move? (the robot should tell how far it moves, how fast.)

c).  give a square. test how far from one point to another.

d). Non-functional safety requirement.

robot lose connection /



Q7.

a). Review: Is it well organized? Changed the idea?

b). correct: ③ get feedback ① product → ② demonstrate

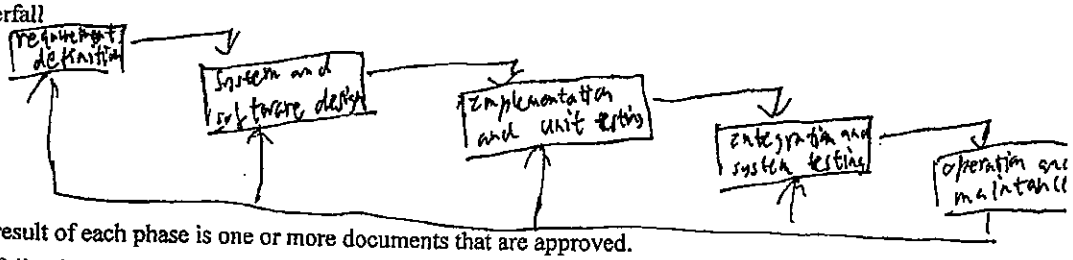
consistency. trackability requirement. ~~Comment~~ some requirements

compliance: use cases. analysis requirements: explain

cht

Software Process Model

Waterfall



The result of each phase is one or more documents that are approved.
The following phase should not start until the previous phase has finished. Problems are left for later resolution, ignored or programmed around.

--Short comes:
This premature freezing of requirements may mean that system won't do what the user wants. It may also lead to badly structured systems as design problems are circumvented by implementation tricks.

--Advantages:

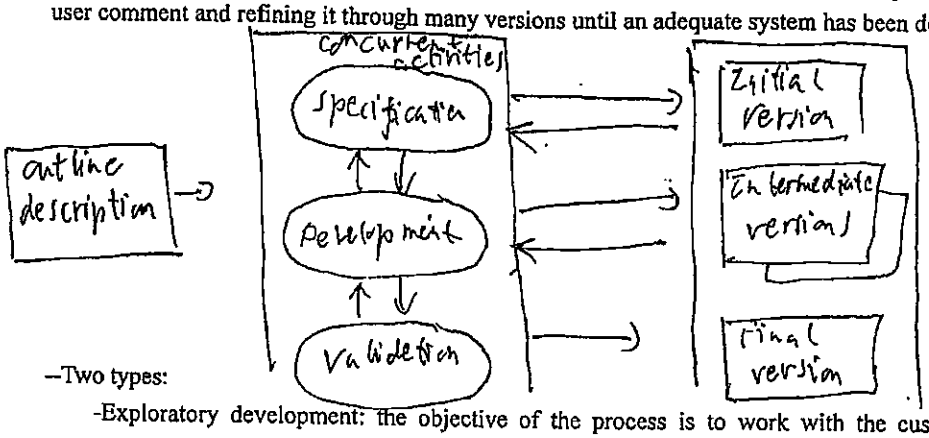
- Documentation is produced at each phase and that it fits with other engineering process models.
-It is major problem is its inflexible partitioning of the project into distinct stages. Commitments must be made at an early stage in the process, which makes it difficult to respond to changing customer requirements.

--Acceptance:

- The requirements are well understood and unlikely to change radically during the system development.
- Software process based on this approach are still used for software development, when the software project is part of a larger systems engineering project.

Evolutionary Model

Evolutionary model is based on the idea of developing on initial implementation, exposing this to user comment and refining it through many versions until an adequate system has been developed.



--Two types:

-Exploratory development: the objective of the process is to work with the customer to explore their requirements and deliver a final system. The development starts with the parts of the system that are understood. The system evolves by adding new features proposed by the customer.

-Throwaway prototyping: the objective of the process is to understand the customer's requirements and hence develop a better requirements definition for the system. The prototype concentrates on experimenting with the customer requirements that are poorly understood.

--Advantage:

-often more effective than waterfall

-specification can be developed incrementally. As user develop a better understanding of their problem, this can be reflected in the software system.

--Short comes:

-process is not visible. Managers need regular deliverable to measure progress. It systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.

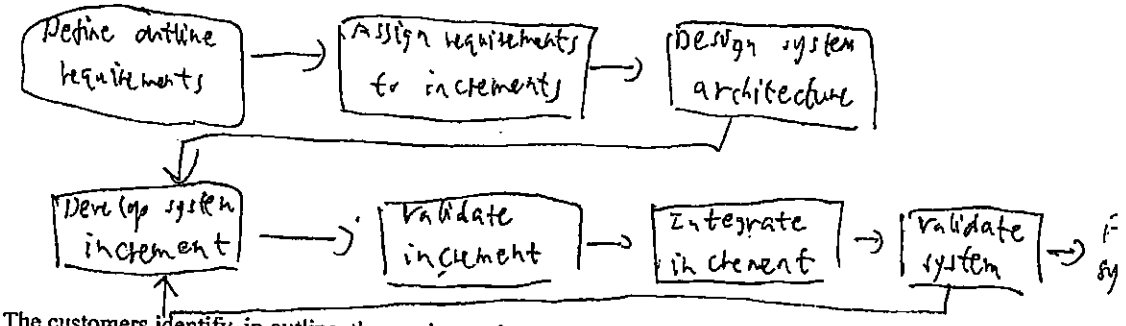
-system are often poorly structured. Continual change tends to corrupt the software structure. Incorporating software changes becomes increasingly difficult and costly.

--Acceptance

-small and medium-sized system.

Incremental delivery

Incremental delivery is an in-between approach that combines the advantages of these models.



The customers identify, in outline, the services to be provided by the system which of the services are most important and which are least important.
A number of delivery increments are then defined.

--Advantage

-customers do not have to wait until the entire system is delivered before they can gain value from it. The first increment satisfies their most critical requirements so they can use the software immediately.

-Customers can use the early increments as prototypes and gain experience that informs their requirements for later system increments.

-There is a lower risk for project failure. Although problems may be encountered in some increments it is likely that some will be successfully delivered to the customer.

-As the highest priority services are delivered first, and later increments are integrated with them, it is inevitable that the most important system services receive the most testing. This means that the customers are less likely to encounter software failures in the most important parts of the system.

--Short comes:

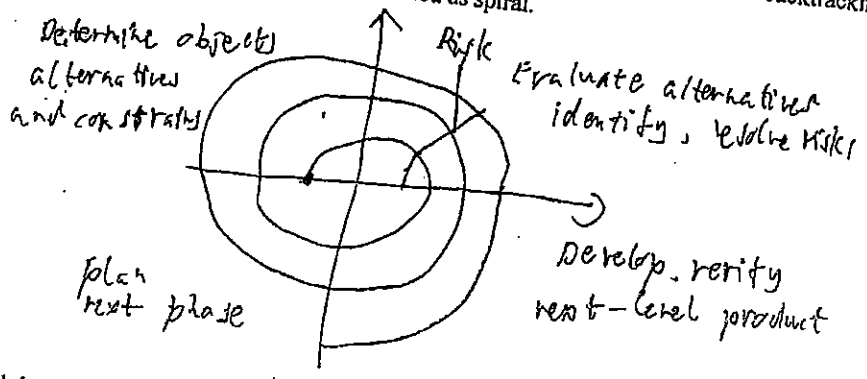
-Increments should be relatively small and each increments should deliver some system functionality.

-Difficult to map the customer's requirements onto increments of the right size.

-Most systems requires a set of basic facilities that are used by different part of the system.

Spiral development

Rather than represent the software process as a sequence of activities with backtracking from one activity to anther, the process is represented as spiral.



Each loop in the spiral represents a phase of the software process.
The innermost loop might be concerned with system feasibility, the next loop with requirements definition, the next loop system design so on.
It explicit recognition of risk in the spiral model.

Advantage:
The model is risk oriented – allowing you to reduce risks to a acceptable level.

Short comes:
The model is complex and requires effective management.
No fixed phases such as specification or design – loops in the spiral are chosen depending on what is required.
It can be difficult to determine when you have addressed the risks to an acceptable level.
Can get caught up in the spiral of risk reduction and never leave to start development.

Risk Management
-People risk
-Product risk
-Business risk

Process

Risk Identification: Technology risks
People risks
Organizational risks
Tools risks
Requirements risks
Estimation risks

Risk analysis: consider each identified risk and make a judgment about the probability and the seriousness.

Risk planning: identifies strategies to manage the risk.

Risk Monitoring: regularly assess each of the identified risk to decide whether or not that risk is becoming more or less probable and whether the effects of the risk have changed.

Requirements:

Functional requirements: what should do? Statements of the services that the systems must provide or are descriptions of how some computations must be carried out.

Non-functional requirements: How to do? Constrain the system being developed and the development process that should be used.

Requirements discovery:

-View points:

It recognizes multiple perspectives and provides a framework for discovering conflicts in the requirements proposed by different stakeholders.

-Interview:

In this interview, requirements engineering team puts question to stakeholders about the system that they use and the system to be developed.

(Question & Answer)

Interview is not an effective technique for eliciting requirements about organizational requirements and constraint.

-Scenarios

Relate to real-life example than to than to abstract descriptions.

Can be particularly useful for adding detail to an outline requirements description.

-Use Case

A scenario-based technique, identify the individual interaction with the system.

Scenarios and use cases are effective technique for eliciting requirements for interavtor viewpoints, where each type of interaction can be represented as a use case.

System models

-Behavioral models:

Data-flow → showing how data is processed by system.

State machine → showing how a system responds to internal or external events.

Data models:

ERA modeling—Entity-Relation-Attribute modeling.

-show data entity, their associated attributes and relationship between the entities.

-because of the explicit typing and recognition of sub and super type, it is also straightforward to implement these models using OO database.

-lack of detail. Should maintain more detailed descriptions of entities, relationships and attributes.

Object models:

Inheritance models – the most general object classes at the top of the hierarchy.

Object aggregation – an object is an aggregate of a set of other objects

Object behavior modeling – to model the behavior of objects, show how the operations provided the objects are used.

Architectural Design:

-System organization

The organization of a system reflects the basic strategy that is used to structure a system.

The repository model:

Two ways – shared data is held in a central database/repository and many be accessed by all sub-systems.

-Each sub-system maintains its own database and passes data explicitly to other sub-system.

-Advantage: Efficient way to share large amounts of data among sub-systems.

Sub-system needs not be concerned with how data is produced.

Centralized management e.g backup, security

Sharing model is published as the repository schema.

-Disadvantage: Sub-system should agree on a repository data model. Inevitably a compromise.

Data evaluation is difficult and expensive.

No scope for specific management policies.

Difficult to distribute efficiently.

Client-server model

A distributed system model which shows how data and processing is distribute across a range of components

Set of stand-alone servers which provide specific services.

Set of clients which call on these services.

Network which allows clients to access servers.

-Advantage: Distribution of data is straightforward.

Make effective use of networked systems. May required cheaper hardware.

Easy to add new servers or upgrade existing servers.

-Disadvantage: No shared-data model, so sub-systems are using different data organization.

Data interchange maybe inefficient.

Redundant management in each server.

No central register of names and servers – it may be hard to find out what servers and services are available.

Layered model(Abstract Machine)

Used to model the interfacing of sub-systems.

Support the incremental development of sub-systems in different layers. When a layer interface changes, only the adjacent layer is affected.

-Advantage: When layer interfaces change or new facilities are added to layer, only the adjacent layer is effected.

Easier to provide multi-platform implementations of an application system.

Only the inner machine dependent layers need be reimplementation.

-Disadvantage: Structuring system in this way can be difficult

Performance can also be a problem because of the multiple levels of command interpretation that are some times required.

Modular decomposition styles

Object Oriented decomposition

Structure the system into a set of loosely coupled objects with well-defined interfaces.

OO decomposition is concerned with identifying object classes, their attributes and operations. When implemented, needs some control model to coordinate the object operation.

-Advantage: Objects are loosely coupled so their implementation can be modified without affecting other objects.

The objects may reflect real world, easy to understand.

Can be reused.

OO implementation languages are widely used.

-Disadvantage: Object interface changes may cause problems.

Complex entities may be hard to represent as objects.

Function-oriented pipelining

Functional transformations process their inputs to produce outputs
May referred to as apipe and filter model(UNIX shell)

Advantage: Support transformation reuse
Intuitive organization for stakeholder communication.
Easy to add new transformations
Relatively simple to implement as either a concurrent or sequential system.

Disadvantage: Require a common format for data transfer along the pipeline and difficult to
Support event-based interaction
Not really suitable for interactive systems.

Control Style

Centralized control

One subsystem has overall responsibility for control and starts and stops other sub-system.

Call-back model

Maybe used at the module level to control functions or objects.

The rigid and restricted nature of this model is both a strength and a weakness. It is Strength because it is relatively simple to analyze control flows and work out how the system will respond to particular inputs . It is a weakness because exceptions to normal operation are awkward to handle.

Applicable to sequential systems

Manager model

Used in real-time system, do not have very tight time constraints.
Applicable to concurrent systems.

Event-driven system

Driven by externally generated events where the timing of the event is out of the control of the sub-system which process the event.

Broadcast model

An event is broadcast to all sub-system. Any sub-system which can handle the event may do so.

Interrupt-driven model

Used in real-time systems.
Interrupts are detected by an interrupt handler and pass to some other component for processing.

Quality Management

Quality assurance – The establishment of a framework of organizational procedures and standard that lead to high-quality software.

The process of defining has software quality can be achieved and how development organization knows that the software has the required level of quality.

Quality planning –includes: product introduction
Product plans
Process descriptions
Quality goals
Risks and risk management

Quality control

Involves monitoring the software development process to ensure that quality assurance procedures and standard are being followed.

-Quality reviews where the software, its documentation and the processes used to produce that software are reviewed by a group of people.

-Automated software assessment where the software and the documents that the produced are processed by some program and compared to the standards that apply to that particular development project.