2008 Paper

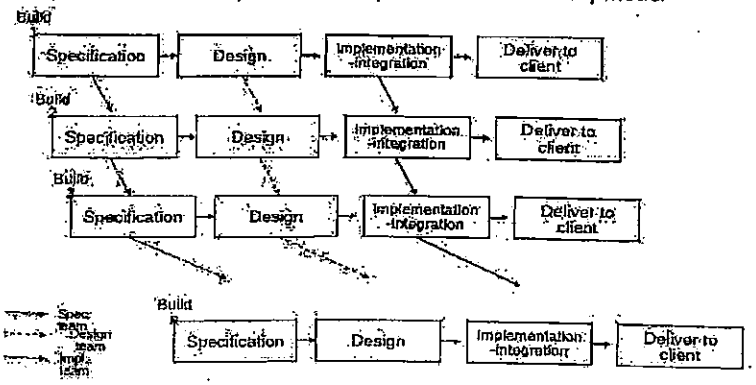
Question 1

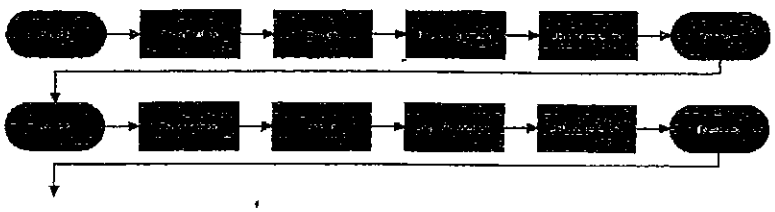
a) Description:

- Group Number 05 – Nexus Software
- The SE process model used was the Evolutionary Model because
  1. Specification and development stages are interleaved
  2. Allows exploratory development
  3. User develops better understanding of the problem, this can reflected in the software system
  4. Based on the idea of developing an initial implementation, exposing this to user comment and refining this through many versions until an adequate system has been developed
  5. Rather than have separate specs, development and validation activities, these are carried out concurrently with raped feedback across the activities

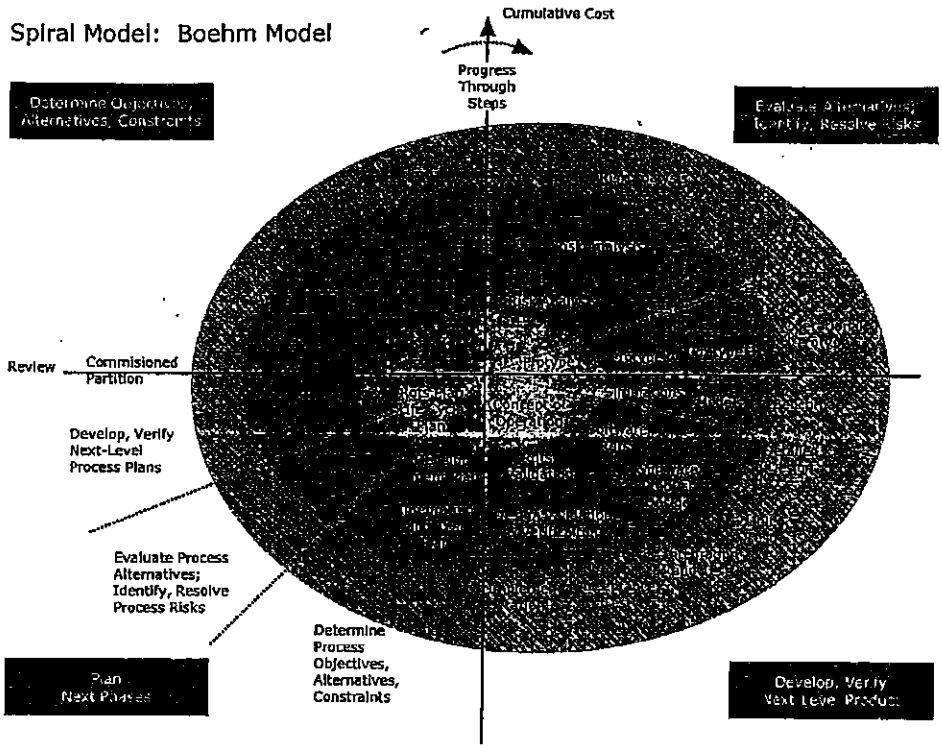
b) Reasons for using this model:

- I would suggest the use of a Evolutionary Model in conjunction with the Spiral model
- Allows the group to work closely with the client to evolve a system from initial requirements
- Allowed the use of rapid prototyping to get client feedback every week, thus allowed the group to either throw away the prototype or develop more
- More effective than Waterfall model in producing systems which meet the immediate requirements of customers
- Allows each milestone / prototype to have risk assessed
- Every 2 weeks would represent one revolution of the spiral model
- Every 2 weeks would represent one layer of the evolutionary model





Spiral Model: Boehm Model



c)

- 4 Reasons for using this model for this project
- Allows for client feedback into the feedback loop
- Allows the client to gather confidence in the team by seeing gradual progression o f the system
- Prototyping may be used in one spiral to resolve requirements uncertainties and hence reduce risk
- On a full scale system developing rapid prototyping is not cost effective and a risk of running out of resources

- Process is not visible – Managers need regular deliverables to measure progress, if systems are developed quickly, it is not cost effective to produce documents which reflect every version of the system

Question 2

a) Reasons for Eclipse:

a) *Makefile*:

```
javaFile = javaFileOne.java javaFileTwo.java
cc = gcc
flag = -o
bin = ./build

minutes: minutes.tex
	pdflatex  minutes.tex;  rm *.log *.toc *.out *.aux; mv *.pdf  ${bin}

buildJava: ${javaFile}
	javac ${javaFile}; mv *.class ${bin}

buildC: main.c modules.c
	${cc} ${flag} main main.c;${cc} ${flag} modules modules.c; mv main
${bin};mv modules ${bin}

compile: buildJava buildC
run:compile
	cd ${bin};java javaFileOne; java javaFileTwo; ./main; ./modules

clean:
	rm –f ${bin}

all: run clean
```

antfile:

```
<?xml version="1.0"?>
<project name="prac2" default="compile" basedir=".">
    <target name="compile" description="compile all the java file">
        <mkdir dir="bin" />
        <javac srcdir="." destdir="bin" />
    </target>
    <target name="run" depends="compile" description="run the program">
        <java classpath="./bin" fork="true" classname="ShapeDriver" />
    </target>
</project>
```

- **Avoidance strategies:** Replace the potential defective GPS unit with a new one from another manufacturer who has a known reliability
- **Minimization strategies:** Write functions which are independent of GPS unit, to allow for physically calculating the position and facing direction by robot itself. Hence, when the results are different, we can know there is an error with the GPS system and can fix it.
- **Contingency plans:** add some function to the robot allowing for immediately stop it while some errors found. Hence the robot won't be damaged and can be fixed.
- **Risk Inductor:** Late delivery of hardware, many reported technology problem

Question 5

a) Host controller: Client-server model (This model is consist of a set of stand-alone servers which offer services to other sub-systems, a set of clients that call on the services offered by servers, and a network which allows the clients to access these services)
   - allow multi robot working together(easy to add new servers)
   - we can upgrade the robot side software but not keep affecting the host side
   - Make effective use of network systems

b) Robot: Repository model (In this model, all shared data is held in a central database that can be accessed by all sub-systems. A system model based on a shared database is sometimes called a repository model)
   - The robot have a large data to share among different components of robot
   - Activities such as backup, security, access control and recovery from error are centralized.(allow for Data analysis)
   - Easy to integrate new components to the robot

c) Additional architecture: Layered model (This model is used to model the interfacing of subsystems. It organizes the system into a set of layers, each of which provide a set of services.)
   - Used to model the interfacing of subsystem
   - Organize the system into a set of layers, each of which provide a set of services
   - Supports the incremental development of subsystems in different layers, when layer interface changes, only the adjacent layer is affected.
   - Performance can be a problem

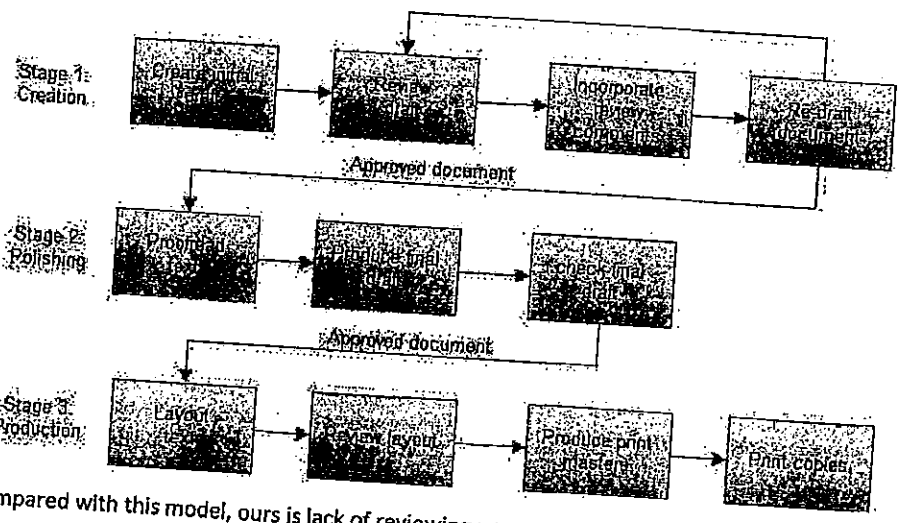
Question 6

a) Technique for clarifying requirements
   1. Observation

2. Interview
3. Scenario
4. Rapid prototyping

Question 7

a) Process: Creation → Polishing → Production
(Drafting, checking, revising, redrafting is an iterative process)



Compared with this model, ours is lack of reviewing process, only one group member has reviewed the document, which is quite not enough. That's why it appears spelling errors in some of the documents.
However, each document was drafted first and then got comments from clients. Add some more details to produce the second draft. That's what we did well.

b) Requirements engineering process:

- Consistent:-**Automated consistency analysis** (If the requirements are expressed as a system model in a structured of formal notation then CASE tools may be used to check the consistency of the model) To check the consistency, the CASE tool must build a requirements database and then, using the rules of the method or notation, check all of the requirements in this database. A requirements analyzer produces a report of inconsistencies which it has discovered)
- Complete: -**prototyping** (In this approach to validation, an executable model of the system is demonstrated to end-users and customers. They can experiment with this model to see if it meets their real needs)