If software were as unreliable as economic theory, there wouldn't be a plane made of anything other than paper that could get off the ground.

Jim Fawcette

Table of Contents

Table of Figures

Question 1: Software lifecycle management

Part a.

Answer:
Keeping in mind the requirements scenario, my choice for the software development cycle would be the incremental model.

Looking at the following ***concerns of the client***:

- Testing out and introducing a new technology for his customers, an embarrassing failure could cost him his reputation.
- Security and privacy are his major concerns.
- He has a big budget.
- He wants a flexible system for any future changes

Reviewing the above shown concerns by the clients makes me choose the incremental model.

The reason why ***v-model won't be a better choice*** are:

- Time taken
- The customer is involved after the end product is completed
- No planning for upgrades to the system
- Risks are not analyzed during the life cycle.

The ***incremental model*** can be used to better go about the development life cycle because:

- It is easier and less costly to make any additions to the scope or the requirements.
- Series of upgrades can be planned
- Risk management becomes easier as they are identified and handled during its iteration
- Every increment has an operational functionality can be shown to the client for feedback
- Minimizes any unpleasant surprise resulting from imposing an entirely new product on the client
- There is a working system at all times.
- Good project visibility
- The artifacts are more stabalized due to the client feedback
- Reduced complexity of testing and debugging
- Each iteration is a well managed milestone
- Extensive work decreases risk

Part b.

Answer:

The Boehm spiral model has the following activities:

For each cycle go through these activities

- Define objectives, alternatives, constraints
- Evaluate alternative, identify and resolve risks
- Develop, verify prototype
- Plan next "cycle"

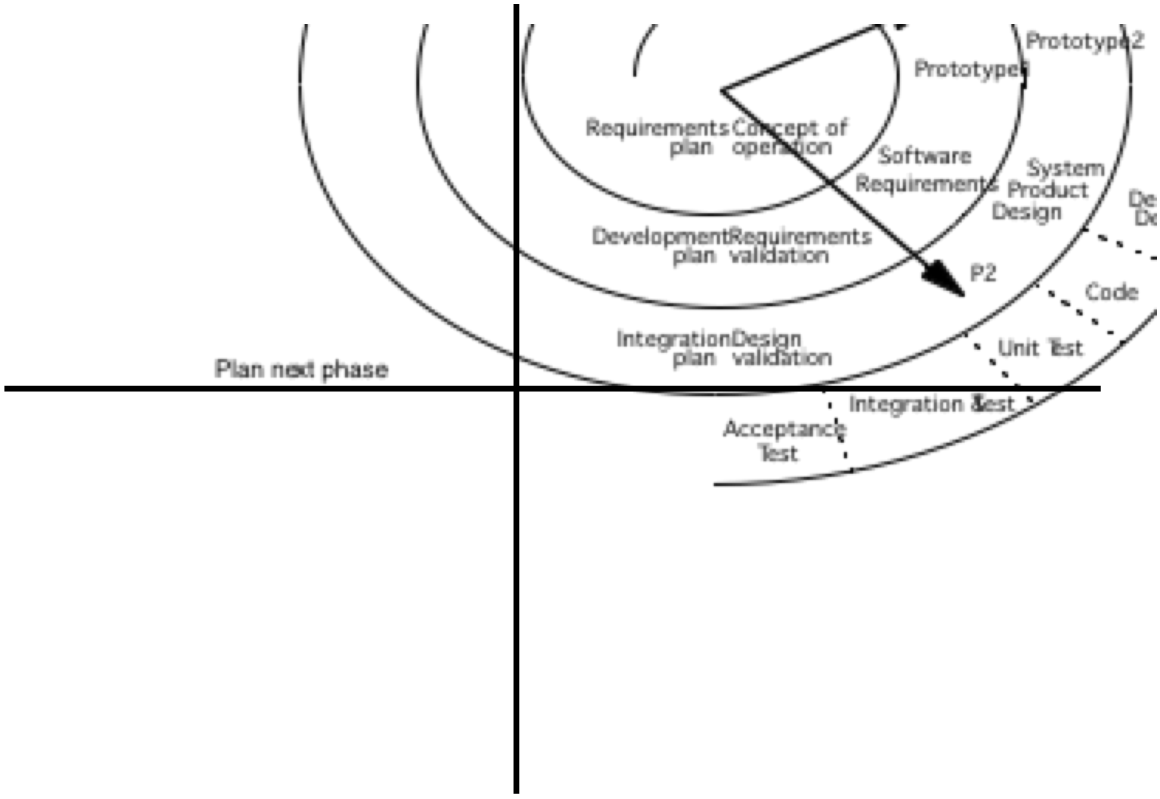

Figure 1 - Boehm Spiral Model

The incremental model can also be represented as a spiral. The spiral size corresponds with system size, and the distance between the coils of the spiral shows resources. This view illustrates the fact that resources can be kept constant while the available functionality grows. Spiral model is a kind of incremental development model with specific focus on risk analysis. In which each trip around the spiral produces something new but this model is considerably more formalized than the classic incremental model.

It's a sort of curled-up incremental waterfall with prototyping thrown in. Spiral development is a process characterized by repeating a set of activities and making improvements between each iteration.

The incremental approach and spiral model can be combined with ease:

Each cycle of the incremental approach develops a specific function, by merging spiral approach each cycle would have following activities:

1. Define objectives, alternatives, constraints
    - Defining requirements
    - Validating them against customer's feedback
    - Multiple design of the subsystem are created
    - Any constraints and dependencies are identified
2. Evaluate alternative, identify and resolve risks
    - Each alternative is tested for any risks
    - Risks are identified
    - Corrective actions for risks are formulated
    - Alternative with least risk is selected
3. Develop, verify prototype
    - The sub system/functionality is being developed and verified using analysis & design
    - Client is shown the developed prototype for validation
4. Plan next "cycle"
    - Next cycle is planned using client's feedback

The spiral model has the same advantages as the incremental approach has, with the following added benefits:

- It explicitly incorporates planning, thus fixing one of the problems of the classic incremental model.
- It explicitly incorporates risk analysis.
- It makes no distinction between development and maintenance (i.e., maintenance is just another phase).
- It treats testing as a risk which must be addressed.
- Emphasis on alternatives and constraints supports the reuse of existing solutions.
- Estimates (budget and schedule) get more realistic as work
- Progresses, because important issues are discovered earlier.

Part c.

Answer:

<u>Identified Risks</u>

Risk		Stage	Probability (H, M, L)	Impact (H, M, L)	Resolution
#	**REQUIREMENTS**				
1	Changing, incomplete or incorrect requirements		H	M	Incremental development: changes can be deferred to later increments
2	Unrealistic deadlines might be planned because of budget or time constraints		H	M	Meticulous cost and schedule estimation and Increment development
#	**DESIGN**				
1	Design will not map with the actual requirements		H	M	Validation and testing of design against requirements specification
2	The skills needed might not be available		M	M	Staffing with top talent, teambuilding, job matching, morale building, cross training, pre-scheduling key people
3	Staff changes will create miscommunication		H	M	Following the incremental mode, the artifacts are more stabalized due to the

				provision of feedback throughout the whole development cycle.
#	**CODING/IMPLEMENTATION**			
1	Defects/bugs will arise	H	M	At every increment, it is easier to trace out and less costly to fix bugs due to reduced complexity of the domain problem.
#	**TESTING**			
1	Adequate testing is not done, thus results in unknown bugs	H	H	Every increment contains a decomposed sub problem of the main problem so it's easier to test.
2	Testing might not be sufficient for the client-side	H	H	As testing is done at each increment and client is involved throughout so it minimizes any unpleasant surprise resulting from imposing an entirely new product on the client by providing a progressive

				introduction in shape of each iteration's end result.
3	More bugs means more time needed on fixing them, this will cause delay in deadlines	H	H	Using incremental model, the end result of each increment is well managed, controlled and tested, which takes care of bugs tracing and fixing in a careful manner for the iterations.
4	In the process of fixing the traced bugs, new ones appear	H	H	The benefit of incremental model is that each iteration can be handled easily as a well managed milestone
#	**DELIVERY/SUPPORT**			
1	System is not according to customer needs	H	H	There is a good project visibility as the client is able to see the system before the end product is completed and give his response.

Note: Key for probability and impact is given in Appendix A

Risk Management Plan

As mentioned by Boehm, a software risk management plan can be constituted by following the steps given below:

1. Identify project's top 10 risks.
2. Present a plan for resolving each risk item.
3. Update list of these risk items, plan and results at each iteration and for each project. (A comparison with previous results can provide a good judgement)
4. Initiate appropriate corrective actions.

By following this type of strategy in the incremental model, risk management becomes easier because risky pieces are identified and handled during its iteration and this comprehensive model thus decreases risk.

APPENDIX A

Key for probability and impact risk management

KEY	
PROBABILITY	
Very likely (H)	High chance of this risk occurring, thus becoming a problem > 70%
Probable (M)	Risk like this may turn into a problem once in a while {30% < x < 70%}
Improbable (L)	Not much chance this will become a problem {0% < x < 30%}
IMPACT	
Catastrophic (H)	Loss of system; unrecoverable failure of system operations; major damage to system; schedule slip causing launch date to be missed; cost overrun greater than 50% of budget
Critical (M)	Minor system damage to system with recoverable operational capacity; cost overrun exceeding 10% (but less than 50% of planned cost
Marginal (L)	Minor system damage to project; recoverable loss of operational capacity; internal schedule slip that does not impact launch date cost overrun less than 10% of planned cost

References

[B]

[Boehm, 1988]
Boehm, B. W. "*A Spiral Model of Software Development and Enhancement."* TRW Defense Systems Group. [May, 1988], pp. P 61-72

[P]

[Parekh, 2005]
Parekh, N. "*Spiral Model - A New Approach Towards Software Development*". [1/13/2005] Available on World Wide Web at: <http://www.buzzle.com/editorials/1-13-2005-64082.asp>

[S]

[Sharp et al, 1999]
Sharp, H; Finkelstein, A & Galal, G. *"Stakeholder Identification in the Requirements Engineering Process"*. [11/12/99]

[Sommerville, 2004]
Sommerville, I
Software Engineering
Chapter 7