Appendix. Questions

- 01. Introduction
- 02. Project Management
- 03. Use Cases
- 04. Domain Models
- 05. Testing
- 06. Design By Contract
- 07. Formal Specifications
- 08. Software Architecture
- 09. Quality Control
- 10. Software Metrics
- 11. Refactoring
- 12. Conclusion

This page is intentionally left blank

1. Introduction (i)

You should know the answers to these questions:

- How does Software Engineering differ from programming?
- Why is programming only a small part of the cost of a "real" software project ?
- Give a definition for "traceability".
- What is the difference between analysis and design?
- Why is the "waterfall" model unrealistic? Why is it still used?
- What's the relationship between iterative development, incremental development and (evolutionary) prototyping?
- How do you decide to stop in the spiral model?
- How do you identify risk ? What do you do when projecting a risk? Which risks do you assess as critical?
- List the 6 principles of extreme programming.
- What is a "sprint" in the SCRUM process ?
- Draw a UML class diagram modeling marriages in cultures with monogamy (1 wife marries 1 husband), polygamy (persons can be married with more than one other person), polyandry (1 woman can be married to more than one man) and polygyny (1 man can be married to more than one woman).
- Draw a UML diagram that represents an object "o" which creates an account (balance initially zero), deposits 100$ and then checks whether the balance is correct.

1. Introduction (ii)

Can you answer the following questions?

- What is your preferred definition of Software Engineering? Why?
- Why do we choose "Correctness" & "Traceability" as evaluation criteria? Can you imagine some others?
- Why is "Maintenance" a strange word for what is done during the activity?
- Why is risk analysis necessary during incremental development?
- How can you validate that an analysis model captures users' real needs?
- When does analysis stop and design start?
- When can implementation start?
- Can you compare the Unified Process and the Spiral Model ?
- Can you explain the values behind the Agile Manifesto ?
- Can you identify some synergies between the techniques used during extreme programming ?
- Is it possible to apply Agile Principles with the Unified Process ?
- Did the UML succeed in becoming the Universal Modeling Language ? Motivate your answer.

2. Project Management (i)

You should know the answers to these questions

- Why is it necessary to define tasks small?
- What is a milestone? What can you use them for?
- What is a critical path? Why is it important to know the critical path?
- What can you do to recover from delays on the critical path?
- How can you use Gantt-charts to optimize the allocation of resources to a project?
- What is a "Known kown", and "Unknown known" and an "Unknown Unknown" ?
- How do you use PERT to calculate the risk of delays to a project?
- Why is it necessary to apply earned value analysis during project management?
- Why does replacing a person imply a negative productivity?
- What's the difference between the 0/100; the 50/50 and the milestone technique for calculating the earned value ?
- Why shouldn't managers take on tasks in the critical path?
- How can you ensure traceability between the plan and the requirements/system ?

You should be able to complete the following tasks

- draw a PERT Chart, incl. calculating the critical path and the risk of delays
- draw a Gant chart, incl. allocating and optimizing of resources
- draw a slip line and a timeline

2. Project Management (ii)

Can you answer the following questions?

- Name the various activities covered by project managament. Which ones do you consider most important? Why?
- Compare PERT-charts with Gantt charts for project planning and monitoring.
- How can you deal with "Unknown Unknows" during project planning ?
- Choose between managing a project that is expected to deliver soon but with a large risk for delays, or managing a project with the same result delivered late but with allmost no risk for delays. Can you argue your choice ?
- Describe how earned-value analysis can help you for project monitoring.
- Would you consider bending slip lines as a good sign or a bad sign? Why?
- You're a project leader and one of your best team members announces that she is pregnant. You're going to your boss, asking for a replacement and for an extension of the project deadline. How would you argue the latter request?
- You have to manage a project team of 5 persons for building a C++ compiler. Which team structure and member roles would you choose? Why?

3. Use Cases (i)

You should know the answers to these questions

- Why should the requirements specification be understandable, precise and open ?
- What's the relationship between a use case and a scenario ?
- Can you give 3 criteria to evaluate a system scope description ? Why do you select these 3 ?
- Why should there be at least one actor who benefits from a use case ?
- Can you supply 3 questions that may help you identifying actors? And use cases?
- Which two basic rules apply to the project plan negotiation ? Why ?
- Which sections should be included into the requirements specification at the end of the inception phase ?
- What's the difference between a primary scenario and a secondary scenario ?
- What's the direction of the <<extends>> and <<includes>> dependencies ?

You should be able to complete the following tasks

- Write a requirements specification (incl. scope definition, actors & use cases and project plan) for your year project.

3. Use Cases (ii)

Can you answer the following questions?

- Can you explain the difference between the two definitions for use cases ?
- Why do use cases fit well in an iterative/incremental development process ?
- Why do we distinguish between primary and secondary scenarios ?
- Assume that you work for a company that does not want to apply use cases in their requirements specification. Which principles would you still apply regardless of the use case notation ?
- What would you think would be the main advantages and disadvantages of use cases ?
- How would you combine use-cases to calculate the risky path in a project plan ?
- Do use-cases work well with agile methods ? Explain why or why not.

4. Domain Modeling (i)

You should know the answers to these questions

- Why is it necessary to validate and analyze the requirements?
- What's the decomposition principle for functional and object-oriented decomposition?
- Can you give the advantages and disadvantages for functional decomposition? What about object-oriented decomposition?
- How can you recognize "god classes"?
- Why do you want to identify objects, responsibilities and collaborations?
- What is a responsibility? What is a collaboration?
- Can you supply 3 issues to take care of when chairing a brainstorm meeting?
- What do feature models define ?
- How does domain modeling help to achieve correctness? Traceability?

You should be able to complete the following tasks

- Apply noun identification & verb identification to (a part of) a requirements specification.

4. Domain Modeling (ii)

Can you answer the following questions?

- How does domain modeling help to validate and analyze the requirements?
- What's the problem with "god classes"?
- Why are many responsibilities, many collaborators and deep inheritance hierarchies suspicious?
- What is the solution for the "2 string puzzle" (see Creative Thinking on page 19)? Why is it a good example of creative thinking?
- Can you explain how role-playing works? Do you think it helps in creative thinking?
- Can you compare Use Cases and CRC Cards in terms of the requirements specification process?
- Do CRC cards yield the best possible class design? Why not?
- Why are CRC cards maintained with paper and pencil instead of electronically?
- What would be the main benefits for thinking in terms of "system families" instead of "one-of-a-kind development ? What would be the main disadvantages ?

5. Testing (i)

You should know the answers to these questions

- What is (a) Testing, (b) a Testing Technique, (c) a Testing Strategy
- What is the difference between an error, a failure and a fault?
- What is a test case? A test stub? A test driver?
- What are the differences and similarities between basis path testing, condition testing and loop testing?
- What are the differences and similarities between unit testing and regression testing?
- How do you know when you tested enough?
- What is Alpha-testing and Beta-Testing? When is it used?
- What is the difference between stress-testing and performance testing?

You should be able to complete the following tasks

- Complete test cases for the Loop Testing example (Loop Testing on page 19).
- Rewrite the binary search so that basis path testing and loop testing becomes easier.
- Write a piece of code implementing a quicksort. Apply all testing techniques (basis path testing, conditional testing [3 variants], loop testing, equivalence partitioning) to derive appropriate test cases.

5. Testing (ii)

Can you answer the following questions?

- You're responsible for setting up a test program. To whom will you assign the responsibility to write tests ? Why ?
- Explain why basis path testing, condition testing and loop testing complement each other.
- When would you combine top-down testing with bottom-up testing? Why?
- When would you combine black-box testing with white-box testing? Why?
- Is it worthwhile to apply white-box testing in an OO context?
- What makes regression testing important?
- Is it acceptable to deliver a system that is not 100% reliable? Why?

6. Design by Contract (i)

You should know the answers to these questions

- What is the distinction between Testing and Design by Contract? Why are they complementary techniques?
- What's the weakest possible condition in logic terms? And the strongest?
- If you have to implement an operation on a class, would you prefer weak or strong conditions for pre- and postcondition? And what about the class invariant?
- If a subclass overrides an operation, what is it allowed to do with the pre- and postcondition? And what about the class invariant?
- Compare Testing and Design by contract using the criteria "Correctness" and "Traceability".
- What's the Liskov substitution principle? Why is it important in OO development?
- When is a pre-condition reasonable?

You should be able to complete the following tasks

- What would be the pre- and post-conditions for the methods top and isEmpty in the Stack specification? How would I extend the contract if I added a method size to the Stack interface?
- Apply design by contract on a class Rectangle, with operations move() and resize().

6. Design by Contract (ii)

Can you answer the following questions?

- Why are redundant checks not a good way to support Design by Contract?
- You're a project manager for a weather forecasting system, where performance is a real issue. Set-up some guidelines concerning assertion monitoring and argue your choice.
- If you have to buy a class from an outsourcer in India, would you prefer a strong precondition over a weak one? And what about the postcondition?
- Do you feel that design by contract yields software systems that are defect free ? If you do, argue why. If you don't, argue why it is still useful.
- How can you ensure the quality of the pre- and postconditions ?

7. Formal Specification (i)

You should know the answers to these questions

- What does it mean when we say that a formal specification is (a) consistent, (b) complete, and (c) unambiguous ?
- What does it mean for a state-chart to be (a) consistent, (b) complete, and (c) unambiguous?
- What does it mean for a algebraic specification to be (a) consistent and (b) complete?
- Why is an UML class diagram a semi-formal specification?
- Can you give 3 arguments against formal methods? Can you counter them?
- What's the distinction between a semi-formal and a formal specification?
- In the code of the binary-search routine. If we replace the < in line 10 with an >, can you prove what is wrong with the program?
- In the discussion of Consistent / Complete / Unambiguous on page 22, can you verify that the addition of the axiom implies inconsistency? And can you show that the removal of the axiom implies incompleteness?

You should be able to complete the following tasks

- Extend the Z specification of the storage tank to include a "full" indicator, indicating when a Storage_Tank is close to capacity.

7. Formal Specification (ii)

You should be able to complete the following tasks (cntnd.)

- Provide the sequence charts for the primary and secondary scenarios of the "Place Order" use case in the chapter on use case. Derive the corresponding state-chart for the class "Order".
- Given a Z or OCL specification, derive a test model using condition testing.
- Given a statechart specification, derive a test model using path testing.

Can you answer the following questions?

- Why is it likely that you will encounter formal specifications?
- Explain the relationship between "Design By Contract" on the one hand and "Input/ Output Specifications", "Algebraic Specifications", "Logic-Based Specifications" and "State based specifications" on the other hand.
- Explain the relationship between "Testing" on the one hand and "Logic-Based Specifications" and "State based specifications" on the other hand.
- Why is it necessary to complement sequence diagrams with state charts?
- What does Cleanroom Development have to do with Formal Specifications?
- You are supposed to build an e-commerce system for selling books over the world-wide web. You must guarantee 24 hours operations and secure transactions. Your boss asks you to look into formal specs; which ones would you advise and why?

8. Software Architecture (i)

You should know the answers to these questions

- What's the role of a software architecture ?
- What is a component ? And what's a connector ?
- What is coupling ? What is cohesion ? What should a good design do with them ?
- What is a pattern ? Why is it useful for describing architecture ?
- Can you name the components in a 3-tiered architecture ? And what about the connectors ?
- Why is a repository better suited for a compiler than pipes and filters ?
- What's the motivation to introduce an abstract factory ?
- Can you give two reasons not to introduce an Adapter (Wrapper) ?
- Assume the ODBC example after applying the bridge pattern (see Solution: Bridge on page 30). Would it be a good idea for the PrintingOdbc to take advantage of special printing features provided by the Oracle database ? Why ?
- What problem does an abstract factory solve ?
- List three tradeoffs for the Adapter pattern.
- What's the distinction between a package diagram and a deployment diagram ?
- Define a sensitivity point and a tradeoff point from the ATAM terminology.

You should be able to complete the following tasks

- Take each of the patterns and identify the components and connectors. Then assess the pattern in terms of coupling and cohesion. Compare this assessment with the tradeoffs.

8. Software Architecture (ii)

Can you answer the following questions ?

- What do architects mean when they say "architecture maps function onto form" ? And what would the inverse "map form into function" mean ?
- How does building architecture relate to software architecture? What's the impact on the corresponding production processes?
- Why are pipes and filters often applied in CGI-scripts ?
- Why do views and controllers always act in pairs ?
- Explain the sentence "Restricts communication between subject and observer" in the Observer pattern
- Can you compare a bridge with an adapter as a way to build a layered architecture ?
- Can you explain the difference between an architecture and a pattern ?
- Explain the key steps of the ATAM method ?
- How would you organize an architecture assessment in your team ?

9. Quality Control (i)

You should know the answers to these questions

- Why is software quality more important than it was a decade ago ?
- Can a correctly functioning piece of software still have poor quality ? Why ?
- If quality control can't guarantee results, why do we bother ?
- What's the difference between an external and an internal quality attribute ? And between a product and a process attribute ?
- What's the distinction between correctness, reliability and robustness ?
- How can you express the "user friendliness" of a system ?
- Can you name three distinct refinements of "maintainability" ? What do each of these names mean ?
- What is meant with "short time to market" ? Can you name 3 related quality attributes and provide definitions for each of them ?
- Name four things which should be recorded in the review minutes ?
- What's the relationship between ISO9001, CMM standards and an organization's quality system ? How do you get certified ?
- Can you name and define the 5 levels of CMM ?
- Can you define a Key Process Area (KPA) ?

9. Quality Control (ii)

You should be able to complete the following tasks

- Given a piece of code and a coding standard, review the code to verify whether the standard has been adhered to.

Can you answer the following questions ?

- Given the Quality Attributes Overview table, argue why the crosses and blanks occur at the given positions.
- Why do quality standards focus on process and internal attributes instead of the desired external product attributes ?
- Why do you need a quality plan ? Which topics should be covered in such a plan ?
- How should you organize and run a review meeting ?
- Why are coding standards important ?
- What would you include in a documentation review checklist ?
- How often should reviews by scheduled ?
- Could you create a review check-list for ATAM ?
- Would you trust software from an ISO 9000 certified company ? And if it were CMM ?
- You are supposed to develop a quality system for your organization. What would you include ?

10. Software Metrics (i)

You should know the answers to these questions

- Can you give three possible problems of metrics usage in software engineering ? How does the measurement theory address them ?
- What's the distinction between a measure and a metric ?
- Can you give an example of a direct and an indirect measure ?
- What kind of measurement scale would you need to say "A specification error is worse than a design error" ? And what if we want to say "A specification error is twice as bad as a design error ?"
- Explain the need for a calibration factor in Putnam's model.
- Fill in the blanks in the following sentence. Explain briefly, based on the Putnam's model.
  + If you want to finish earlier (= decrease scheduled time), you should ... the effort ... .
- Give three metrics for measuring size of a software product.
- Discuss the main advantages and disadvantages of Function Points.
- What does it mean for a coupling metric not to satisfy the representation condition ?
- Can you give 3 examples of impreciseness in Lines of Code measurements ?
- What's the difference between "Mean time to failure" and "Average time between failures" ? Why is the difference important ?

You should be able to complete the following tasks

- Given a set of use cases (i.e. your project) calculate the use case points.

10. Software Metrics (ii)

Can you answer the following questions ?

- During which phases in a software project would you use metrics ?
- Why is it so important to have "good" product size metrics ?
- Can you explain the two levels of calibration in COCOMO (i.e. C & S vs. M) ? How can you derive actual values for these parameters ?
- Can you motivate why in software engineering, productivity depends on the scheduled time ? Do you have an explanation for it ?
- Can you explain the cone of uncertainty ? And why is it so relevant to cost estimation in software projects ?
- How can you decrease the uncertainty of a project bid using Putnam's model ?
- Why do we prefer measuring Internal Product Attributes instead of External Product Attributes during Quality Control ? What is the main disadvantage of doing that ?
- You are a project manager and you want to convince your project team to apply algorithmic cost modeling. How would you explain the technique ?
- Where would you fit coupling/cohesion metrics in a hierarchical quality model like ISO 9126 ?
- Why are coupling/cohesion metrics important ? Why then are they so rarely used ?
- Do you believe that "defect density" says something about the correctness of a program ? Motivate your answer ?

11.Refactoring (i)

You should know the answers to these questions:

- Can you explain how refactoring differs from plain coding?
- Can you tell the difference between Corrective, Adaptive and Perfective maintenance? And how about preventive maintenance ?
- Can you name the three phases of the iterative development life-cycle? Which of the three does refactoring support the best? Why do you say so?
- Can you give 4 symptoms for code that can be "cured" via refactoring?
- Can you explain why add class/add method/ass attribute are behaviour preserving ?
- Can you give the pre-conditions for a "rename method" method refactoring ?
- Which 4 activities should be supported by tools when refactoring?
- Why can't we apply a "push up" to a method "x()" which accesses an attribute in the class the method is defined upon (see Refactoring Sequence (3/5) on page 24)?

You should be able to complete the following tasks

- Two classes A & B have a common parent class X. Class A defines a method a() and class B a method b() and there is a large portion of duplicated code between the two methods. Give a sequence of refectorings that moves the duplicated code in a separate method x() defined on the common superclass X.
- What would you do in the above situation if the duplicated code in the methods a() and b() are the same except for the name and type of a third object which they delegate responsibilities too?

11.Refactoring (ii)

Can you answer the following questions?

- Why would you use refactoring in combination with Design by Contract and Regression Testing?
- Can you give an example of a sequence of refactorings that would improve a piece of code with deeply nested conditionals?
- How would you refactor a large method? And a large class?
- Consider an inheritance relationship between a superclass "Square" and a subclass "Rectangle". How would you refactor these classes to end up with a true "is-a" relationship? Can you generalise this procedure to any abusive inheritance relationship?

12. Conclusion (i)

You should know the answers to these questions

- Name 3 items from the code of ethics and provide a one-line explanation.
- If you are an independent consultant, how can you ensure that you will not have to act against the code of ethics ?
- What would be a possible metric for measuring the amount of innovation of a manufacturing company ?
- What can we do to avoid that glue-code erodes our component architecture ?
- Explain in 3 sentences how a software tool could recommendation on certain fields in a bug report (severity, assigned-to, estimated time).
- Which components would be more susceptible to defects: the ones with a high level of ownership or the ones with a low level of ownership ?

When you chose the "No Silver Bullet" paper

- What's the distinction between essence and accidents ?
- Name 3 reasons why the building of software is essentially a hard task? Provide a one-line explanation.
- Why is "object-oriented programming" no silver bullet ?
- Why is "program verification" no silver bullet ?
- Why are "components" a potential silver bullet ?

When you chose the "Killer Robot" paper

- Which regression tests would you have written to prevent the "killer robot" ?
- Was code reviewing applied as part of the QA process ? Why (not) ?
- Why was the waterfall process disastrous in this particular case ?
- Why was the user-interface design flawed ?

12. Conclusion (ii)

Can you answer the following questions?

- You are an experienced designer and you heard that the sales people earn more money than you do. You want to ask your boss for a salary-increase; how would you argue your case ?
- Software products are usually released with a disclaimer like "Company X is not responsible for errors resulting from the use of this program". Does this mean that you shouldn't test your software? Motivate your answer.
- Your are a QA manager and are requested to produce a monthly report about the quality of the test process. How would you do that ?
- Give 3 scenarios to illustrate why it would be interesting to know which developers are most experienced with a give piece of code.
- It has been demonstrated that it is feasible to make reliable recommendations concerning fields in a bug report. What is still needed before such a recommendation tool can be incorporated into state-of-the art tools (bugzilla, jira, …)

When you chose the "No Silver Bullet" paper

- Explain why incremental development is a promising attack on conceptual essence. Give examples from the different topics addressed in the course.
- "Software components" are said to be a promising attack on conceptual essence. Which techniques in the course are applicable ? Which techniques aren't ?

When you chose the "Killer Robot" paper

- Recount the story of the Killer Robot case. List the three most important causes for the failure and argue why you think these are the most important.