Study guide (for the exam)

I. Oliveira, v2020-01-13

Use this study guide as a review list of the learning objectives for the written exam. It does not provide a list of possible questions, rather a clarification of the subjects to be covered.
This is mainly a compilation of the learning objectives stated for each TP lesson.

What is included in the SDLC?

The work of the Analyst in the development team

Main references: TP-A2

- Explain what is the Systems Development Life Cycle
- Describe the main activities (key steps) within each of the four phase of the SDLC
- Describe the role and responsibilities of the Analyst in the SDLC
- Identify and present distinctive features of CASE tools

The Unified Process/OpenUP

Main references: TP-A2, TP-B3b

- Describe the structure of the OpenUP (phases and iterations)
- Describe the lifecycle objectives of each phase
- Identify key modeling/development activities associated to each phase
- Is OpenUP an agile method?

Key characteristics of Agile methods

Main references: TP-D1, TP-D2, TP-D3

- Identify the distinctive characteristic of sequential processes, such as the waterfall approach.
- Identify the distinctive practices of Agile methods (what is new in the process model, comparing to the "traditional" approach?).
- Elaborate on the argument that "The waterfall approach tends to mask the real risks to a project until it is too late to do anything meaningful about them."
- Identify advantages of structuring a project in iterations, producing increments.
- Characterize the principles of backlog management in Agile projects.
- Given a "principle" (defined in the Agile Manifesto), explain it in your own words, focusing on the its novelty (with respect to "classical" approaches) and impact/benefit.

The role of (visual) modeling

Main references: TP-A3

- Justify the use of models in systems engineering
- Describe the difference between functional models, static models and behavior models.
- Enumerate advantages of visual models
- Explain the organization of the UML (classification of the diagrams)
- Identify the main diagrams in UML and their modeling viewpoint
- Read and create Activity Diagrams, Use Case Diagrams, Class Diagrams, Sequence Diagrams, State Machine Diagrams, Deployment Diagrams, Package and Component Diagrams.

Analysis models

Requirements engineering practices

Main references: TP-B1, TP-B2

- Distinguish between functional and non-functional requirements
- Distinguish between usage-centric and product-centric approaches to requirements elicitation
- Identify, in a list, instances of business rules, functional requirements and quality attributes.
- Justify that "requirements elicitation is more than requirements gathering"
- Identify well-written and ill-written requirements (follow or not the S.M.A.R.T. criteria)
- Elaborate on the fact that requirements elicitation is an intense human interaction challenge (identify risks, challenges…)
- Make informed recommendations of requirements elicitation techniques for different types of projects
- Distinguish between Vision and SRS and their role in the requirements engineering
- Define and identify quality attributes

- Explain what is requirements traceability and, in particular, the use of a requirements traceability matrix to relate requirements and use cases
- Enumerate key features of Requirements Management software tools/environments.

Modeling the domain/business context

Main references: TP-B4a, TP-B4b
Describing the business concepts

- Draw a simple class diagram to capture the concepts of a problem domain.
- Present two strategies to systematically uncover candidate concepts to include in a domain model.
- Spot implementation-specific constructs that may pollute the domain model.
- Describing the business processes
- Read and draw activity diagrams do describe organization/business workflows.
- Identify the proper use of actions, control flow, object flow, events and partitions with respect to a given process description.
- Relate the "business concepts" (classes in the domain model) with object flows in the activity models.

Functional Modeling with use cases

Main references: TP-B3a.

- Describe the process used to identify use cases.
- Read and create Use Case Diagrams.
- Review existing use case models to detect semantic and syntactic problems.
- Describe the essential elements of an use case specification.
- Explain the complementary use of use-case diagrams, activity diagrams, and use case narratives.
- Elaborate on the meaning of "use-case driven development".
- Explain the six "Principles for Use-Case Adoption" proposed by Ivar Jacobson (with respect to "Use Case 2.0")
- Understand the relation between requirements and use cases
- Identify the requirements related disciplines and activities in the OpenUP

Structural Modeling

Main references: TP-B4*, TP-C1

- Distinguish between top-down algorithmic systems analysis and domain-abstractions based.
- Justify the use of structural models in systems specification.
- Explain the relationship between class and object diagrams
- Critically review existing models to capture the domain concepts.
- Review a give class model for syntax and semantic problems, given a problem description.
- Describe the types and roles of the different associations in the class diagram.
- Identify proper use of the association, composition and aggregation to model the relationship between cooperating objects.
- Identify the proper use of association classes.

Behavior Modeling

Main references: TP-B5

- Explain the role of behavior modeling in the SDLC
- Understand the rules and style guidelines for sequence, communication and state diagrams
- Understand the complementarity between sequence and communication diagrams
- Map sequence diagrams in object-oriented code and reverse.
- Critically review existing sequence diagrams models to describe the cooperation between devices or software entities.

Design & implementation models

System architecture views

Main references: TP-C4.

- Explain the activities associated with software architecture development.
- Identify the abstract elements of a software architecture
- Identify the layers and partitions in a layered architecture software architecture.
- Critically review an existing package diagram to illustrate a logical architecture
- Critically review an existing component diagram to describe the tangible parts of software
- Critically review an existing deployment diagram to describe the installation of a system

Classes and methods design (developer perspective)

Main references: TP-C1, TP-C4, TP-B4

- Explain how the use case model can be used to drive the design activities
- Explain the principles of low coupling and high cohesion in OO design
- Critically review an existing OO design w. r. t. appropriate level of coupling/cohesion
- Enumerate and describe the GRASP principles (Larman)
- Enumerate and describe the SOLID principles (R. Martin)
- Give examples of OO designs that break the SOLID principles.
- Explain the code implications of the navigability modeled in the class diagram
- Construct a class diagram and a sequence diagram given a Java code.
- Define software pattern and describe the elements of a software pattern.
- Explain the 3 categories of patterns; recognize/give examples for each category.

Selected practices in the software construction

Quality assurance

Main references: TP-E1.

- Identify validation and verification activities included in the SDLC
- Describe what are the layers of the test pyramid
- Describe the object of unit, integration, system and acceptance test
- Explain the lifecycle of TDD
- Describe the "debug-later" and "test-driven" approaches, as seen by J. Grenning.
- Explain how the QA activities are inserted in the development process in a classical approach and in Agile methods
- What is the V-Model?
- Relate the story acceptance criteria with Agile testing.

Continuous delivery

Main references: TP-E2

- Distinguish the concepts of Continuous Integration, Continuous Delivery and Continuous Deployment.
- Describe the main features of a Continuous Integration platform/service.
- In the context of CI, what are the steps included in the "build process"?
- What is the meaning of the practice "Make Your Build Self-Testing", as recommended by M. Fowler?

Complementary approaches

User-stories in the agile methods

Main references: TP-D3, TP-E1.

- Define stories and give examples.
- Elaborate on why is the "Sticky note" metaphor (for planning and control) so common in Agile projects.
- Identify the key elements of a "persona".
- How are the "personas" used in user-stories approaches?
- Compare user stories and use cases with respect to commonalities and differences.
- Compare "Persona" with Actor with respect to commonalities and differences.
- Critically review the acceptance criteria part of a user story.
- What are story points and how are they assigned?
- Describe the concepts of velocity and burnup chart (as used in PivotalTracker).
- Elaborate on whether use-cases and user-stories are redundant or complementary approaches (should stick with one approach? In which conditions?...)
- Relate the concepts of use-cases, user-stories and use-case slices as described in the Use-Case 2.0 paper (by Ivar Jacobson).

User-centered design

Main references: Invited Talk (by Samuel Silva from IEETA-Research).

- Explain the user centered design (UCD) iterative approach (process).
- Distinguish between "usability" and "user experience" w.r.t. a software system.
- Explain how the UCD approach improves the user experience.
- Compare UCD and use-case driven development with respect to commonalities and differences.

Customer Journey Maps

Main references: Invited Talk (by Cátia Oliveira, from AlticeLabs).

- Explain the elements included in a CJM.
- Compare CJM and use-case modelling with respect to commonalities and differences.

SCRUM for agile teams

Main references: TP-D2; invited Talk (by Luciano & Ana, from Nokia Networks).

— Identify advantages of structuring a project in iterations, producing increments.
— Characterize the principles of backlog management in Agile projects.
— Identify the roles in a SCRUM team and the key “ceremonies”

Additional resources

See also:

— Sample written test structure.

Questions & doubts

— Post in the #ams_ect channel (https://detiuaveiro.slack.com).