Repetition Questions (dt. Wiederholungsfragen) Lesson 14

Topics and Concepts (Link to Lecture, via Fact Sheets)

The lesson of the lecture today covered the following concepts (see fact sheets in script folder):

1. Architectural Evaluation
2. Partial recapitulation (dt. Wiederholung) of selected topics/templates from previous lectures.

Questions

Topic/Concept: Architectural Evaluation

1. What is the purpose of architectural evaluation?
2. How does architectural evaluation relate to the other two architecting phases?
3. Discuss a tradeoff between two quality attributes in a Web shop.
4. What does ATAM stand for?
5. List two key concepts in ATAM.

Topic/Concept: Entire Lecture (Cross-Cutting Questions)

1. Why should NFRs be specified in a SMART way?
2. What happens to the candidate components identified ad hoc or with the help of the heuristic from lesson/exercise 4 later on a project (e.g., during a contruction phase)?
3. List at least three architectural decisions to be made in Solution Strategy.
4. Discuss the relationship between DDD and architectural decision making in a few sentences.
5. What is the difference between a managed container and a library? List at least two.
6. Which JEE and Spring concepts are suited to implement core tactic DDD patterns?
7. Which concepts were introduced in lecture lesson 7 (on November 1, 2017)?
8. What is the difference between a bounded context and a subdomain in DDD?
9. Which two general concepts are specifically relevant when defining SOA from an architect's perspective?
10. Describe the ESB pattern from at least two of the 4+1 viewpoints (e.g. logical, process).
11. What is the relationship a) between the ESB pattern and the API Gateway pattern? b) SOA and microservices? c) SOA and REST?
12. Which ASRs are particularly relevant for the decision for or against Event Sourcing and CQRS? List three.
13. What is the main driver for IT architecture according to lecture 13 and how do quality attributes fit in?
14. Describe the relationship between business objectives, design goals, architectural principles and patterns, architectural decisions, and technology/platforms (hint: have a look at the slides of lecture 11).

Answers

Topic/Concept: Architectural Evaluation

1. *Assess and communicate whether the design is able to address the requirements (NFRs in particular) with the objective to reduce risk of budget overruns (e.g., time, money): "Am I doing the right things? Am I doing things right?" (in terms of decision making)*

2. *It studies the artifacts produced in the previous two phases and assesses their suitability and "fitness" w.r.t. the business objectives and project goals.*
3. *e.g., security vs. usability and security vs. performance because encryption algorithms need input (e.g., keys) and processing capacity.*
4. *Architecture Tradeoff Analysis Method*
5. *Quality attributes, architectural decisions (not surprisingly); architectural decisions also discussed under sensitivity points and tradeoff points*

Topic/Concept: Entire Lecture (Cross-Cutting Questions)

1. *Because only specific and measurable NFRs can be reviewed properly and tested against. A fuzzy NFR such as "The system should be very user friendly" will not get you far in synthesis and evaluation (i.e., not suited to serve as decision justification when selecting alternate designs), unless other risk mitigation strategies are chosen (e.g., user community represented in development team, continuous A/B usability testing)*
2. *They can serve as planning items, development specifications, and are the subject of buy vs. build decisions.*
3. *Layering scheme, database paradigm and vendor/product (or open source asset), client-server cut (assignment of layers to tiers). See slides 20 and 21 of lesson 14 for more recurring Architectural Decision (AD) topics.*
4. *DDD proposes patterns, some of which are alternatives to each other, so ADs are required (pattern selection decisions). Furthermore, the DDD (and any other) patterns have to be implemented with the help of technologies and products (or open source assets); after a pattern has been selected, a number of follow on decisions are required, both on the architectural level and then on the implementation level.*
5. *The patterns Inversion of Control and Dependency Injection are commonly used in managed containers but not necessarily in libraries. Availability of management interfaces (command line, graphical). See slide 14 in lesson 5.*
6. *Enterprise JavaBeans (EJBs) and Spring Beans, with annotations such as* `@Autowired` *and* `@Component` *(and/or others). JPA and Spring Data (for some but not all repositories).*
7. *There was no lesson 7 (just a reminder).*
8. *See table on slide 11 in lesson 8; also see integration and collaboration fallacies on slide 17 in lesson 14.*
9. *Architectural principles and patterns.*
10. *Logical components: router, adapter, translator; interactions: integration flow (e.g., specified in a component interaction diagram) would represent dynamic process view. If the ESB software has to be highly available and therefore clustered, the deployment view becomes relevant too (see guest lecture in week 10).*
11. *See exercise 11: same responsibilities, different context (in the text/pictures that we looked at in lecture and exercise), external and client facing (frontend to backend) vs. internal and somewhat centralized (backend to backend).*
12. *Accuracy/consistency; performance (speed of query processing), maintainability (two models/channels), several more. See slides 11 and 14 in lesson 12.*
13. *Complexity and diversity; NFRs differ per application, which makes a city planning level landscape hard to maintain and evolve.*
14. *See slide 13 in lesson 14 (UML class diagram showing a simplified domain model for software architecting).*