Repetition Questions (dt. Wiederholungsfragen) Lesson 8

Topics and Concepts (Link to Lecture)

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

1. Strategic Domain-Driven Design (DDD)
2. Buy-vs.-Build Decision Support

In the corresponding exercise[1], we reviewed a case study that applied these concepts and practiced them.

Questions

Topic/Concept: Strategic DDD

1. What is a Context Map?
2. Which relationships between Bounded Contexts are described as patterns in strategic DDD?
3. Name one commonality and one difference between the Bounded Context and the Subdomain patterns.
4. List the three types (variants) of Subdomains.
5. How do Bounded Contexts and Aggregates relate to each other?
6. What makes Bounded Contexts interesting from a solution strategy point of view and from an integration architect's perspective? Identify at leat two usage scenarios.

Topic/Concept: Buy vs. Build Decision

1. What are the steps in the evaluation process that was introduced in the lecture?
2. Name at least three evaluation criteria, and explain where they come from. It is ok to reference previous lecture topics and concepts.
3. Why might a pure quantitative comparison of Commercially-off-the-Shelf (COTS) products (or other decision alternatives) be misleading in practice?
4. What is the role and usage of DDD in Buy-vs.-Build decision making?

Answers

Topic/Concept: Strategic DDD

1. *A Context Map shows instances of the Bounded Context pattern and their relations, which are typed. See slide 17 of the lecture for the position and role of this pattern in strategic DDD.*
2. *See slides 18 and 19 of the lecture: Published Language, Shared Kernel, Open Host Service (OHS), Customer/Supplier, Conformist, Anti-Corruption Layer (ACL). Additional relationship patterns (that were not explained in detail in the lecture) are Big Ball of Mud, Partnership and Separate Ways (see "Domain-Driven Design Reference" by E. Evans).*
3. *Both are partitioning strategies: top-down from business requirements (subdomains) vs. bottom up from people/team organization and system/project reality (Bounded Context). In reality, they have an n:m relation although 1:1 would be ideal (but only appears in rare "green field" scenarios). See slide 11 of the lecture for a comparison.*

[1] ../3-exercises-solutions/ZIO-AppArch-ExerciseWeek8.pdf

4. *Core Domain, Supporting Subdomain, Generic Subdomains*
5. *Each Bounded Context contains one or more Aggregates (see slides 11 and 15 of the lecture and, optionally, page 58 in "Implementing DDD" book by V. Vaughn)*
6. *Bounded Contexts can suggest units of deployment and distribution, so finding suited ones is a key task in SOA/microservices design and API Design and Management. They can also support project organization (formation of teams and subteams) and COTS software evaluation (as shown in the article "Using domain-driven design to evaluate commercial off-the-shelf software"[2]).*

Topic/Concept: Buy vs. Build Decision

1. *See slide 24: from orientation/criteria definition to weighted criteria scoring (of a long list) to PoT/PoT (for entries in a short list) to SWOT analysis to decision recommendation with rationale.*
2. *License, cost, functional fit, NFRs (the SMART criteria and the FURPS taxonomy can be used here); one should keep the design philosophy of the software vendor/owner in mind, it must be compatible with those of the teams using/customizing/integration the chosen software. See slide 26 of the lecture.*
3. *See slide 25 of the lecture: false sense of accuracy, see thoughts in "A rational design process: how and why to fake it"[3] and cognitive bias discussion. When used, a sensitivity analysis should always be done and one has to make sure not compare apples and oranges (thought experiment: what is the result of this operation:* `0.8*license cost + 0.6*feature set` *and what does it mean?)*
4. *DDD context maps can a) help during orientation and scoping (by setting explicit boundaries) and b) visualize overlap in domain models between COTS product and other systems (e.g., if shared kernels are identified). Such overlap leads to system integration effort because certain subdomains (or parts of them) may have been implemented multiple times; furthermore, the relationship role taken by a COTS product often is OHS which makes the consumer side a conformist that has to be protected by an ACL, which in turn has to be designed, implemented, tested, and maintained. Such integration efforts might overrun the development effort savings from the product purchase.*

[2] http://dddcommunity.org/wp-content/uploads/files/practitioner_reports/landre_einar_2006_part2.pdf
[3] https://www.researchgate.net/publication/260649064_A_Rational_Design_Process_How_and_Why_to_Fake_it