Assignment 3: Couse Project Milestone 3 Derivation and Evaluation of Solution and Deployment Blueprints

Software Architecture
Suzanne Barber

In this assignment, you will work from the business blueprint you derived in Assignment 2 to build the rest of your blueprint family. Upon completion, your family will be composed of a "business" blueprint (BB) and two "deployment" blueprints (DB) (Figure 1). (In this assignment, you do not need to create a separate "solution" blueprint (SB). You can think of it as creating a specific SB for each DB, collapsing the SB into the DB.)

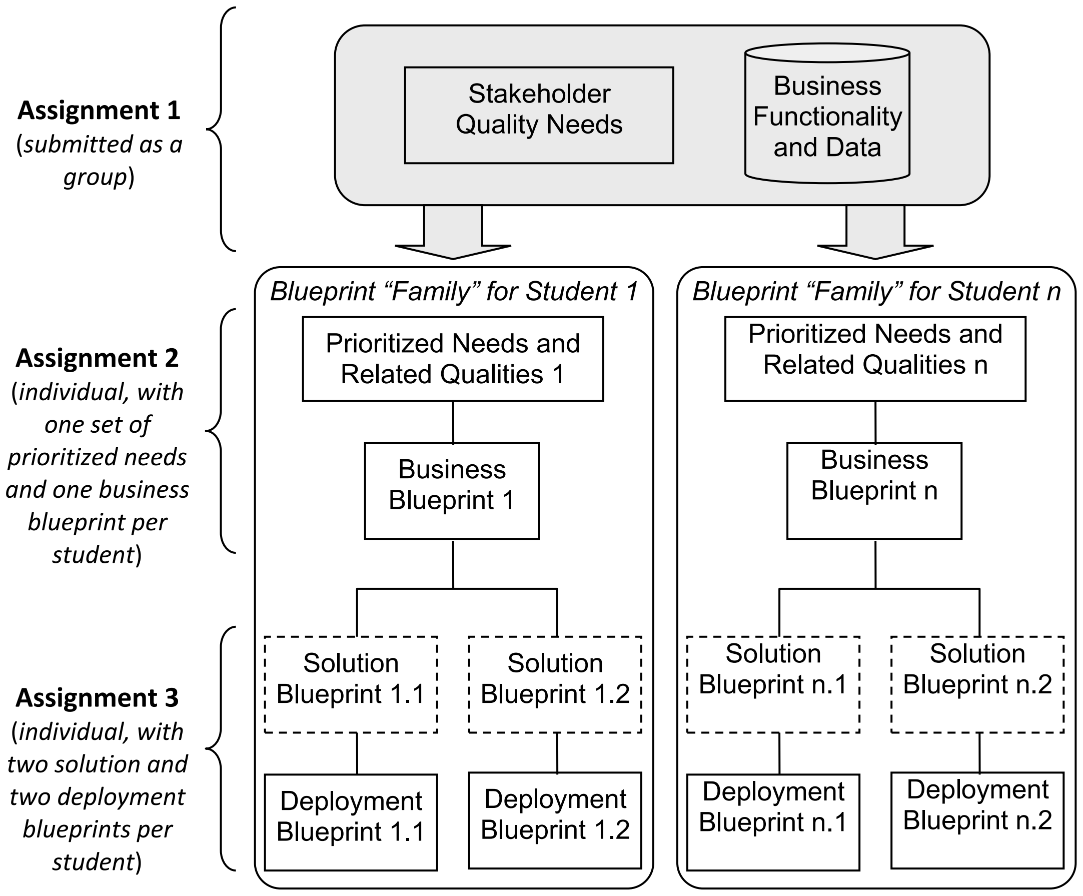
Assignment 1
(submitted as a group)
Assignment 2
(individual, with one set of prioritized needs and one business blueprint per student)
Assignment 3
(individual, with two solution and two deployment blueprints per student)
Stakeholder Quality Needs
Business Functionality and Data
Blueprint "Family" for Student 1
Prioritized Needs and Related Qualities 1
Business Blueprint 1
Solution Blueprint 1.1
Solution Blueprint 1.2
Deployment Blueprint 1.1
Deployment Blueprint 1.2
Blueprint "Family" for Student n
Prioritized Needs and Related Qualities n
Business Blueprint n
Solution Blueprint n.1
Solution Blueprint n.2
Deployment Blueprint n.1
Deployment Blueprint n.2

Figure 1: Blueprint "families" derived from stakeholder needs and domain functionality

Background

- The BB is intended to be the architect's implementation-independent "vision" for how domain functionality should be assigned to system components (i.e., what is the functional and data responsibilities of each component). BB connectors represent the dependencies between components due to the I/O between functions allocated to those components. BB allocation is driven by heuristics that suggest how structural properties, such as coupling, cohesion, and complexity, may promote or inhibit system qualities of interest (i.e., non-functional requirements expressed by stakeholders). Depending on the heuristics applied, the choice of components can be influenced by business areas, organizations, performer roles, legacy implementations, and other factors.
- The SB and DB "instantiate" the BB by describing implementation details; BB components are realized by SB and DB components and connectors that represent software modules, computing platforms, people, connectivity, etc. While the SB and DB are implementation specific, the level of detail between two DBs may be quite different, e.g., identifying off-the-shelf software as a black box versus describing the exchange of control and data between software modules. The SB and DB may be inspired by any design principle that helps satisfy stakeholder needs, including, but not limited to, what are commonly known as architectural styles.
- Regardless of the blueprint type, evaluations are chosen to determine whether *stakeholder needs* have been met based on the *blueprint representation*.
    - Business Blueprint evaluation involves measuring the blueprint's structure (component size and coupling) to determine if heuristics have been applied that yield a structure promoting qualities of interest.
    - Solution Blueprint evaluation involves determining how well solutions (i) satisfy functionality and data in the BB and (ii) match the BB component boundaries.
    - Because the Deployment Blueprint includes implementation details, evaluations are specific to the implementation.

Part 1: Deriving Deployment Blueprints (DB)

Create a *two* Deployment Blueprints (DB) from the Business Blueprint derived in Assignment 2 (See Figure 1).

- *Description:*
    - The DB contains implementation-specific components and connectors that realize BB functionality. The DB may be at any level of detail and DB component boundaries may not map to BB components exactly. In general, DB components will represent some combination of the following:
        - *solutions* (e.g., software, hardware, people) responsible for delivering the functionality and data in your BB. (The SB comprises these solutions.)
        - *environment/infrastructure* (e.g., computing platforms, networks, display devices, databases, system software) required to support the solutions.

- *Deliverables:*
    - <u>Representation (components and connectors)</u>
        - Graphic:
            - As appropriate for selected style(s):
                - Note: Generate the graphic in any tool.
                - Note: The graphic should provide enough detail so that…
                    - the *<u>primary</u> style or design principle* is evident (e.g., if client/server is the primary style among the four you chose [see below], client and server components must be identifiable) and
                    - the *solutions* (e.g., software, hardware, people) responsible for delivering the functionality and data in your BB are identifiable. (The SB comprises these solutions.)
                - Note: Since your graphic reflects a style/design principle, <u>provide (i) a reference with an example of a similar style/design principle graphic and (ii) a relevant excerpt from that reference</u> (e.g., if your blueprint reflects client/server style, provide an excerpt from a web page or other source with a client/server graphic).
        - Text (Word or Excel):
            - Description of each element in graphic (components and connectors).
                - Note: Every "box" and "line" must be described.
            - Mapping from BB components (with respective functions and data) to DB components and connectors.
                - Note: Since the DB includes the solutions in the SB (e.g., software, hardware, people) responsible for

delivering the functionality and data in your BB, discuss the following:

        - Which BB functionality each solution satisfies.
        - Which BB data each solution satisfies.
        - How I/O between functions is satisfied by connections between DB components (e.g., if function1 sends data1 to function2 and they are satisfied by two DB solutions, then there must be connectivity of some form between the two solutions).

- Derivation Rationale
    - Identify 4 architectural styles and/or other design principles that provide the basis for your DB. Include a reference for each.
        - Describe how those styles and principles address stakeholder needs. (Be sure to discuss all stakeholder needs, even ones you were unable to fully satisfy.)
        - Identify potential conflicts and describe tradeoffs based on priorities as well as style/principle effectiveness.
    - If the DB represents a refactoring of BB components, explain how this may sacrifice the BB's "vision" and the and why the refactoring is justified.

Part 2: Evaluating Solution Blueprint Compliance

Calculate the following compliance metrics for the solutions (technologies) in each Deployment Blueprint that are responsible for delivering the functionality and data in your BB.

For each component ***c*** and technology ***t***, calculate ***CompFuncCoeff(c,t)*** and ***CompDataCoeff(c,t)***.

CompFuncCoeff(c,t): Degree to which technology ***t*** complies to functions in component ***c***.

$$CompFuncCoeff(c,t) = \frac{|CompFunc_{regd}(c,t)|}{|CompFunc(c)|}$$

where:

$CompFunc_{regd}(c,t)$	Set of functions in component ***c*** to which technology ***t*** is registered (i.e., capable of providing).
CompFunc(c)	Set of functions defined in component ***c***.

CompDataCoeff(c,t): Degree to which technology ***t*** complies to data in component ***c***.

$$CompDataCoeff(c,t) = \frac{|CompData_{regd}(c,t)|}{|CompData(c)|}$$

where:

$CompData_{regd}(c,t)$	Set of data in component ***c*** to which technology ***t*** is registered.
CompData(c)	Set of data defined in component ***c***.

For each technology ***t***, calculate ***CompFuncBoundaryError(t)***.

CompFuncBoundaryError(t): Degree to which technology ***t*** does not satisfy and exceeds the functionality for the component with which it best complies.

$$CompFuncBoundaryError(t) = \left(1 - \max_{c_{m1} \in BB}(CompFuncCoeff(c_m, t))\right) + \sum_{c \neq c_m} CompFuncCoeff(c, t)$$

where:

$\max_{c_{m1} \in BB}(CompFuncCoeff(c_m, t))$	Maximum *CompFuncCoeff* value for technology ***t*** across all components in the Business Blueprint (BB).
$\sum_{c \neq c_m} CompFuncCoeff(c, t)$	Sum of *CompFuncCoeff* values for technology ***t*** for all other components.

Part 3: Planning Evaluation of Deployment Blueprints Using an ATAM Quality Attribute Utility Tree

Generate an ATAM Quality Attribute Utility Tree for each Deployment Blueprint you defined in Part 1.

- **Level 1: "Utility"**
  - *Define the root node "Utility" and discuss how you would aggregate quality assessments to obtain an overall "goodness" of the system.*
    - Comment: Consider priorities assigned to qualities and what manipulation is needed to normalize results across different types of evaluation.

- **Level 2: Qualities**
  - *Define level 2 nodes for all qualities.*
    - Comment: Create level 2 nodes for all qualities specified in Assignment 2 Part 1.

- **Level 3: Sub-factors**
  - *Define level 3 nodes that break down qualities in level 2.*
    - Comment: Sub-factors may be associated with a heuristic, style, or pattern (e.g., "reducing coupling increases likelihood of reusability") or simply one aspect of the higher-level quality (e.g., "transaction response time" is one consideration when evaluating performance).

- **Level 4: Specific objectives under the sub-factors**
  - *Define level 4 nodes that express objectives under the level 3 sub-factors in the context of the deployment blueprint and prioritize those objectives based on importance and ease by which they can be achieved.*
    - Comment: An example of an objective under sub-factor "transaction response time" might be "Course registration database responds in no more than 3 seconds."
    - Comment: When prioritizing, use a (H)igh, (M)edium, (L)ow scale for each of the following dimensions:
      - Importance of the objective the success of the system.
      - How easy it will be to achieve the objective.
    - Comment: If you were able to obtain the relevant data/statistics, it should be straightforward to determine if the objective has been met.

- **Level 5: Metrics for determining if an objective has been met.**
    - *Define level 5 nodes that state specifically what data/statistics you need to determine if a level 4 objective has been met and how you will obtain those data/statistics. Rate each data/statistic based on the degree to which it accurately predicts/assesses whether the objective has been met.*
        - <u>Comment</u>: An example might be "Activate SQL database profiler and record latencies for transactions X, Y, and Z while executing a test script simulating peak load over a sustained period."
        - <u>Comment</u>: When rating, use a (H)igh, (M)edium, (L)ow scale.
        - <u>Comment</u>: You may reference the coupling/cohesion and size/complexity metrics from Part 1 and the compliance metrics from Part 2.
        - <u>Comment</u>: You may reference your scenarios from Assignment 1. (Not for functional validation, per se, but for a context in which to evaluate an objective.)

Comments

- This is *your* design with *your* chosen domain -- I did my best not to constrain the problem to give you the opportunity to be creative. *This is a research task as much as a specification task.*
- As I mentioned in Assignment 2, you can think of the BB, SB, and DB as your "back of the envelope" or "napkin" architectures: If someone asked you to sketch your architecture with or without implementation details, what would you draw? Each blueprint is *focused on conveying some aspect* of the architecture *and no more* (i.e., promoting separation of concerns). The following summarizes what we're trying to communicate and what the sketch recipient needs to know for each blueprint.
    - The BB representation *only* specifies how responsibility for functionality and data in the domain is allocated to a collection of (abstract) components.
        - What does the BB need to tell the recipient? – The recipient is asking, "How would you recommend I divide up these responsibilities?"
        - Note that the BB itself does not describe *why* you're making that recommendation. That's covered by your derivation plan and accompanying evaluation of BB structure.
    - The SB representation *only* specifies a set of technologies/solutions (e.g., software, hardware, people) and which functionality and data in the BB each technology/solution satisfies.
        - What does the SB need to tell the recipient? – The recipient is asking, "To what degree do these technologies/solutions satisfy the functional and data responsibilities in the BB and do they divide up the responsibilities in the same way?"
        - Note that the SB itself does not describe *why* you chose those technologies/solutions. That's covered by your evaluation of technology/solution compliance to the BB.
    - The DB representation *only* specifies how selected styles/design principles are to be applied for a given deployment.
        - What does the DB need to tell the recipient? – The recipient is asking, "Where do the technologies/solutions in the SB fit in the overall deployment and what are the necessary constraints on that deployment?"
        - Note that the DB itself does not describe *why* you established those constraints. That's covered by your styles and design guidelines.
- Architectural styles are not the only bases for your architecture. You may reference any general principle that promotes the qualities demanded by your stakeholders. Regardless, no derivation decision should be made without such principles.
- Regardless what level of detail you specify in your DB, the DB satisfies the functional requirements in your BB and maps to the BB component boundaries in some way, whether fully respecting them or deviating from them.

- Some examples of styles:
    - Blackboard
    - Client-server
    - Database-centric architecture
    - Distributed computing
    - Event Driven Architecture
    - Peer-to-peer
    - Pipes and filters
    - Representational State Transfer (REST)
    - Call-and-return modularity (including object-oriented programming)
    - Service-oriented architecture
- Some examples of design principles:
    - Network load balancing
    - Replication
    - Redundancy

References

- Bass, Clements, and Kazman, Software Architectures in Practice (1st Edition).
- BinSubaih and Maddock, "Using ATAM to Evaluate a Game-based Architecture", http://www.cs.rug.nl/~paris/ACE2006/papers/BinSubaih.pdf.
- Kazman, Klein, and Clements, "ATAM: Method for Architecture Evaluation," technical report CMU/SEI-2000-TR-004, August 2000, http://www.sei.cmu.edu/pub/documents/00.reports/pdf/00tr004.pdf.