Assignment 2: Couse Project Milestone 2

Derivation and Evaluation of Business Blueprints

Software Architecture

In assignments 2 and 3, you will specify a software architecture (blueprint) "family" based on the domain problem and stakeholder requirements outlined in assignment 1. Each family will be composed of a "business" blueprint (BB), two "solution" blueprints (SB), and two "deployment" blueprints (DB) (Figure 1). Assignment 2 involves derivation and evaluation of a business blueprint, and Assignment 3 builds upon this result with the derivation and evaluation of solution and deployment blueprints.

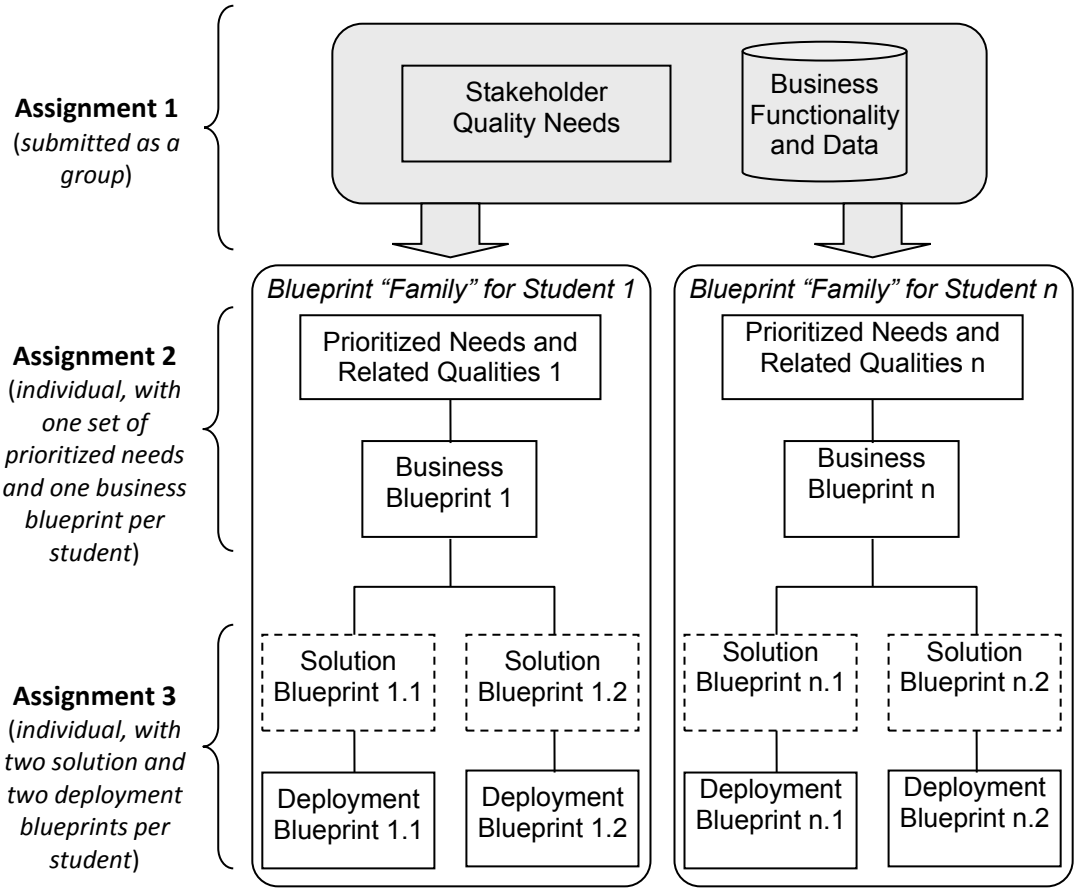

Figure 1: Blueprint "families" derived from stakeholder needs and domain functionality

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 derivation entails making only *structural* decisions with regard to allocating functions and data to components based on non-functional requirements (system qualities). 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.

The business blueprint in Assignment 2 and the solution and deployment blueprints in Assignment 3 are derived from the functional, non-functional, and installation requirements in Assignment 1. While you worked in groups for Assignment 1, you will be working independently for Assignment 2 and Assignment 3. As you derive your blueprints, you may determine that you need to additional detail about the requirements in Assignment 1 (i.e., as though you were asking the requirements engineer to elaborate). You are welcome to revise Assignment 1, *but everyone in your group must agree to the changes and derive their blueprints from the same set of requirements (i.e., everyone must derive their business blueprints from the same revised Assignment 1).*

Part 1: Prioritize Stakeholder Qualities/Constraints and Associate them with Quality Categories

Create a table that enumerates and prioritizes each of the stakeholder needs from Assignment 1 Section 1.5 and classifies them based on architectural quality categories, such as reusability, maintainability, performance, cost, reliability, etc. Note that priorities are often not stated explicitly, requiring you to infer them. Similarly, you may only have priorities in the context of one stakeholder, requiring you to merge priorities across stakeholders.

- *Deliverables:*
    - Table (Word or Excel) with stakeholder need, priority, classification, and a brief justification for priority assignment.
    - Assignment 1 resubmitted (please resubmit Assignment 1 whether revised or not).

Part 2: Derive a Business Blueprint (BB)

Create a BB based on stakeholder needs and assigned qualities and priorities from Part 1. As discussed in class, business blueprint derivation involves collecting domain functions and data into logical components that are "responsible" for those functional requirements.

- *Deliverables:*
    - Representation (components and connectors)

      Graphic
        - UML class model where classes represent components and associations are I/O dependencies between components (annotate the line with the number of *input* dependencies next to the respective component). Use class methods to represent the functions allocated to a component.
            - Note: You need not depict the data allocated to each component.
            - Note: When annotating your component-to-component associations, take into account *all* data/event dependencies between components (see the following items below: *Function-to-function data I/O, Function-to-function event I/O*, and *Function-references-data-stored*).
            - Note: Generate graphics with any UML-capable tool, such as Visio, Argo, Rose, Enterprise Architect, etc.

Text (Word or Excel)

- List of components.
- List of functions allocated to each component.
  - Note: A function can only be allocated to one component (i.e., cannot be replicated across components).
- List of data allocated to each component.
  - Note: *You need not allocate events (or "transient" data elements).* For example, assume event "UserHasBeenAuthenticated" is generated by the "Authenticate" function and required by the "Add Course" function. The event is clearly passed from "Authenticate" to "Add Course" (creating a dependency between them), but it does not need to be allocated to a component because no component has the responsibility to "manage" it.
  - Note: A data element can only be allocated to one component (i.e., cannot be replicated across components).
- List of function I/O dependencies between components resulting from the following:
  - *Function-to-function data I/O:* Function1 is allocated to Component1, Function2 is allocated to Component2, and Function1 generates Data1 that is required by Function2.
  - *Function-to-function event I/O:* Function1 is allocated to Component1, Function2 is allocated to Component2, and Function1 generates Event1 that is required by Function2.
  - *Function-references-data-stored:* Function1 allocated to Component1 uses input Data3 retrieved directly from Component3, where it was allocated, or Function1 allocated to Component1 outputs Data3 that is stored in Component3, where it was allocated.
    - Note: If a data element has been allocated to a component, if must be referenced directly by *some* function in the business blueprint.

- List of function I/O dependencies between components and external producers/consumers resulting from the following:
    - *External-to-function data I/O:* Function1 is allocated to Component1 and requires DataX1 as input from an external producer.
    - *External-to-function event I/O:* Function1 is allocated to Component1 and requires EventX1 as input from an external producer.
    - *Function-to-external data I/O:* Function1 is allocated to Component1 and outputs DataX2 to an external consumer.
    - *Function-to-external event I/O:* Function1 is allocated to Component1 and outputs EventX2 to an external consumer.
- List of function I/O satisfied within the same component:
    - *Function-to-function data I/O:* Both Function2 and Function4 are allocated to Component2, and Function2 generates Data4 that is required by Function4.
    - *Function-to-function event I/O:* Both Function2 and Function4 are allocated to Component2, and Function2 generates Event4 that is required by Function4.
    - *Function-references-data-stored:* Function4 allocated to Component2 inputs or outputs Data5, which is also allocated to Component2.

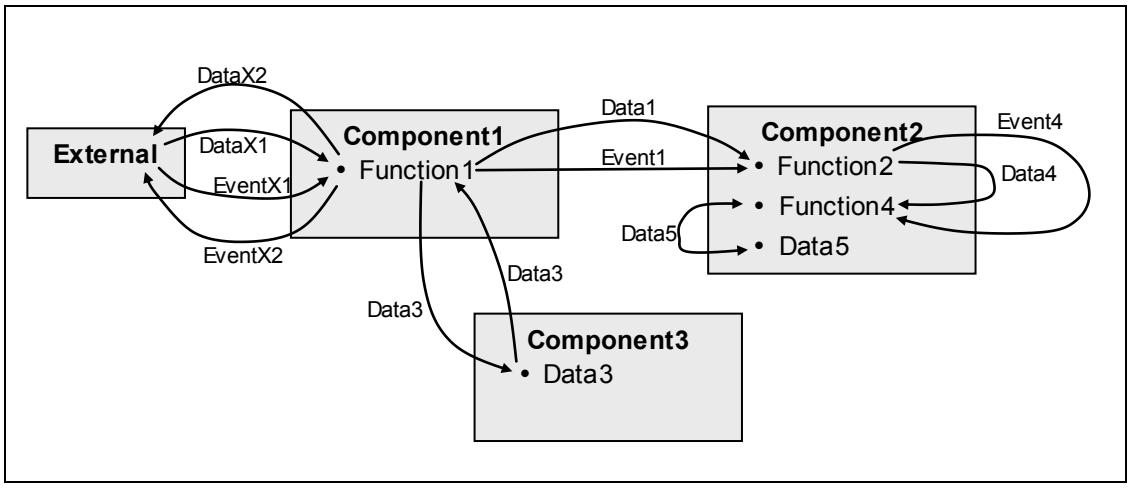

Figure 2: I/O dependency types

- Derivation Plan and Rationale
  - Create a derivation plan composed of at least *5 relevant heuristics from the heuristic guidelines* that accompany this assignment. (At least one of those heuristics must be related to coupling/cohesion or size/complexity in order to complete Part 3.)
    - Associate heuristics with your blueprint goals.
    - Describe how the heuristics map to stakeholder quality needs.
      - Note: Your blueprint goals should help your mapping effort.
    - It is possible that some of your stakeholder needs cannot be addressed at the business blueprint level, but you *must discuss all stakeholder needs and be explicit about those that cannot be addressed*.
    - Prioritize the goals and heuristics according to the priorities identified in Part 1 and your judgment of heuristic effectiveness (e.g., heuristic 1 is more effective than heuristic 2 for promoting reusability because…). Justify your priority assignments.
    - Identify potential conflicts and describe tradeoff options based on priorities and heuristic effectiveness.
    - Describe how the bootstrap you specified was a reasonable starting point based on your derivation plan.

Part 3: Evaluate Business Blueprint Structure

Calculate the following structural metrics for the Business Blueprint from Part 2.

- *Deliverables:*
    - Coupling and Cohesion Metrics
        - Number of Inputs/Outputs between components
            - For each component:
                - Number of inputs (data and events) across all functions in the component that are received from another component + Number of outputs (data and events) across all functions in the component that are sent to another component.
                    - Note: This is essentially any data or event that crosses the component boundary, excluding data/events sent to our received from external.
                    - Note: If the same data or event is received from multiple functions in another component, count each receipt as a separate instance. For example, if Function1.1 and Function1.2 in Component1 both send Data1 to Component2, consider that two inputs/outputs when calculating the "Number of Inputs/Outputs" for Component1 and Component2.
        - Number of dependencies between components
            - For each component:
                - Number of components to which this component sends some output or from which this component receives some input.
        - Degree of Cohesion
            - For each component
                - Percentage of functions in the component that receive all inputs from functions/data within the component and send all outputs to functions/data within the component.
    - Size and Complexity Metrics
        - Number of functions in a component
            - For each component
                - Number of functions allocated to that component.
        - Number of data elements in a component
            - For each component
                - Number of data elements allocated to that component.

- Number of components in the blueprint
    - For the blueprint as a whole
        - Number of components.

- Component complexity
    - For each component
        - Number of data elements + number of functions + number of inputs and outputs for each function.

- Support for Applied Heuristic

  For at least one of the heuristics you chose when deriving your Business Blueprint, discuss how the metrics above can be used to demonstrate that the heuristic has been applied.

Comments

- 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?
    - What are you trying to communicate?
    - What is the recipient of the sketch asking?
- By proposing a suite of blueprints from abstract to more concrete, we are simply formalizing best practice. In other words, as a software engineer you are accustomed to specifying different types of models because you want to (i) ask questions that are more easily answered by one model versus another and (ii) convey information to stakeholders in a clear and concise way. The precise BB, SB, and DB breakdown is not as important as the concept of a hierarchy of architectures. (Note: We use the term "blueprint" because it implies visions and plans.)
- It is not our intention to teach a single architectural approach or modeling notation in this class. Rather, we want the takeaway message to include general principles such as the following:
    - choosing the right representation,
    - systematic, stepwise derivation and evaluation,
    - capturing derivation rationale, and
    - retaining traceability.
- In the spirit of the software engineering mantra that suggests "problems found earlier are cheaper/easier to fix," we recommend you push as much vision as possible "up" to the BB. If a quality *can* be affected by collecting functionality in one way or another, allow that to influence your BB derivation.
- Heuristics, styles, and design principles often conflict, requiring tradeoffs in the BB, SB, and DB. Tradeoffs are made in part based the relative priorities of stakeholder needs and the associated system qualities.