SIoTIP: Assignment Part 2b

Software Architecture: 2016–2017 project assignment

1 Instructions

In this part of the assignment, you will complete your existing initial architecture by (i) revising the decisions made during part 2a (based on the feedback, only if necessary), and (ii) addressing the remaining requirements.

You are free to decide how you will extend this architecture: you can either continue executing ADD as in part 2a, or you can employ a more freestyle architectural design process. You must however adhere to the intrinsic priorities of the requirements presented earlier in the assignment for part 2a, meaning that you should make sure that low-priority requirements are not addressed at the expense of the high-priority ones.

Starting point: Your architecture from Part 2a.

The report

For part 2b, you are only required to document the resulting architecture, i.e., you should not document any intermediate steps you have taken. The resulting architecture should be documented in the following sections (similar to the example report):

* Introduction (optional)

1. Architectural Decisions:
    - **Architectural decisions for each QAS**: Briefly discuss your key architectural decisions for each non-functional requirement. Pay attention to the solutions that you employed (in your own terms or using tactics and/or patterns). Next, provide a detailed discussion of your architectural decisions. Explain your solutions in terms of the components of the client-server view, and discuss the alternatives that you considered (if any). This must be a self-contained and complete explanation of your architecture. Also, pay attention to deployment decisions that are crucial for achieving certain non-functional requirements. Also, discuss any alternative deployments that you considered (if any).
    - **Other decisions**: Do not forget to document any other important architectural decisions you made, which do not fit under the main non-functionals. These should be documented as well, following the same structure as above.
    - **Discussion**: Use this section to discuss your resulting architecture in retrospect. For example, discuss the strong points and the weak points of your architecture. Is there anything you would have done differently with your current experience? Shortly summarize the roles played by the different team members in this section as well (include noteworthy discrepancies in the effort spent between individual team members).
2. Client-server view (UML Component diagram):
    - The context diagram of the client-server view
    - The primary diagram
3. Decomposition view (UML Component diagram): list the decompositions of the components of the client-server view which you have further decomposed.

4. Deployment view (UML Deployment diagram):
   - The context diagram of the deployment view
   - The primary diagram
5. Scenarios: Illustrate how your architecture fulfils the most important interactions. These must at least include:
   - Sensor data being processed by the system (*UC11*).
   - Subscribing to an application (*UC19*).
   - Applications issuing actuation commands (*UC12*).
   - Sensors/actuators failing, causing (i) deactivation of specific applications, (ii) a redundant sensor/actuator to take over in the context of a single application (*UC14*, *Av3*, *UC18*).
   - Application crash (*Av2*).
   - Plugging in a new pluggable device (sensor or actuator) (*U2*, *UC4*).
   - Detection and handling of communication channel failure (*Av1*, *UC15*).
   - Upgrading an application (*UC22*, *U1*).
   - Sending actuation commands via a mobile app (*UC26*, *UC27*, *UC12*).

   Describe these scenarios using UML Sequence diagrams or UML Communication diagrams.
6. Element catalog and datatypes:
   - Element Catalog: List all components and describe their responsibilities and provided interfaces. Per interface, list all methods using a Java-like syntax and describe their effect.
   - Common interfaces: List any common interfaces provided by multiple components.
   - Exception types: List all exception types thrown by your operations.
   - Data types: List and describe all data types defined in your interface specifications.

   Sort all elements alphabetically within their subsections for ease of navigation.

The result should be a readable and self-contained report that systematically lists the most important decisions first and subsequently provides more details. The easiest way to accomplish this is to adhere to the LaTeX template provided to you (cf. below).

Formatting rules You can use the tool of your choice to author the report, but you must deliver a single PDF file, preferably with indexes enabled for easy navigation (easy to achieve in LaTeX with the hyperref package). No other digital formats are accepted. The file must be self-contained (e.g., no extra figures in attachment) and readable (e.g., font size in pictures is adequate). Pages must be numbered. The cover page must mention the course name and the team member names (including their student id numbers). If you decide to use LaTeX, you can use the template that has been provided (on Toledo) and which contains a wide array of additional pointers.

For UML modeling, you are recommended to use *Visual Paradigm for UML* (available in the PC labs, and installable on your own machine).

Delivery The deadline to turn in your report for part 2b is **Wednesday, May 10 (noon)**. You are expected to (i) upload a self-contained PDF document on Toledo, and (ii) deliver a printed copy of this report in project post boxes (in the student room A00.03) on the main floor of the Department of Computer Science.

Good luck! The Software Architecture team.