Remote Measurement, Monitoring and Control System: Assignment Part 2b: Architectural extension

Software Architecture: Project Assignment 2013-2014

In this part of the assignment, you will extend an existing initial architecture and address the requirements that have not yet been addressed. 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: Appendix B[1] presents the initial architecture from which you will start part 2b of the assignment.

Stop condition: The guideline is that you must approximate a decomposition level that is similar to that of the architecture examples presented in the course. The rule of thumb is that you don't decompose further than three levels of decomposition (if such level of detail is required).

1 The report for part 2b.

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 given example report):

1. Introduction (optional)
2. Overview:
   - Architectural decisions: Briefly discuss your 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).
   - 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 otherwise with your current experience?
3. Context diagram (of the component-and-connector view)
4. Main component diagram:
   - The diagram itself and accompanying explanation.
   - Main architectural decisions: provide a detailed discussion of the main architectural decisions (i.e. those for the most important requirements and those that had the most impact). Explain your solutions in terms of the components of the main component diagram, and discuss the alternatives that you considered. This must be a self-contained and complete explanation of your architecture. Imagine you had to describe how the architecture supports these key requirements to someone that is looking at the main component diagram diagram only. Hide unnecessary details (these should be shown in the decompositions).

[1]This starting point represents what could be a result of part 2a in terms of the key decisions that have been taken, but not in terms of the level of detail and depth of documentation.

5. Key decompositions: discuss the decompositions of the components of the main component diagram which you have further decomposed.
6. Deployment view:
   - The context diagram of the deployment view.
   - The main deployment diagram itself and accompanying explanation. Pay attention to the parts of the deployment diagram which are crucial for achieving certain non-functional requirements. Also discuss any alternative deployments that you considered.
7. Scenarios: Illustrate how your architecture fulfills the most important data flows. As a rule of thumb, focus on the scenario of the domain description. Use Chapter 9 of "Software architecture in practice" (*"Documenting Behavior"*). Describe the scenario in terms of architectural components using UML sequence diagrams and further explain the most important interactions in text.
   Illustrating the scenarios serves as a quick validation of the completeness of your architecture. If you notice at this point that for some reason, certain functionality or qualities are not addressed sufficiently in your architecture, it suffices to document this, together with a rationale of why this is the case according to you. You do **not** have to further refine you architecture at this point.
8. 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 and exceptions if any. List all elements and interfaces alphabetically for ease of navigation.
9. Data type catalog: List and describe all data types defined in your interface specifications. List them alphabetically for ease of navigation.

The result should be a readable and self-contained report that systematically lists the most important decisions first and gradually provides more details towards the end.

1.1 Formatting rules

To author the report, you can use the tool of your choice, but you must deliver a single PDF file, possibly 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). Provide an index to your document and allow to navigate all references in the PDF (e.g., use hyperref in LateX). Pages must be numbered. The cover page must mention the course name and the team member names (including their student id numbers). Be aware that failing to comply with these instructions will cost you points. If you decide to use LaTeX, you can use the template that has been provided (on Toledo).

For UML modeling, you are required 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 May 12 (2pm). 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.