LAB 2: Architecture Design

1. OBJECTIVES

1.1 Evaluate architecture alternatives and select candidate architecture
1.2 Design subsystem interface.

2. INTRODUCTION

2.1 In the last Lab session, you performed an initial round of requirements analysis to identify and clarify the requirements assigned to your subsystem. From building the conceptual model, you have identified the Entity classes representing the data that will be input, transformed and output by your subsystem.

2.2 In this Lab, you will operate on a higher level of abstraction by considering how your subsystem needs to interact with other subsystems in order to satisfy the allocated requirements.

2.3 Furthermore, all subsystems will need to work together in an integrated system to meet the overall system design goals e.g. fulfilling specified software quality attributes, non-functional agreements, constraints.

2.4 Lead developers from the various subsystems, armed with an understanding of their subsystem, must come together to generate architecture alternatives to address to overall design problem.

2.5 Evaluate the architectural alternatives by “walking through” key usage scenarios. The objective is to see how well the design alternative supports that scenario. A candidate architecture is selected, with awareness of the trade-off’s involved.

2.6 From the candidate architecture, design the subsystem interfaces according to the relationships and interactions required.

2.7 This process is highly iterative. Be prepared to move back and forth between architectural and subsystem design, between requirements analysis and component interface design.

3. PROCEDURE

3.1 *Evaluate architecture alternatives and select candidate architecture*

3.1.1 Study the system usage scenarios and design goals. With an initial understanding of subsystem functionality, consider interaction between subsystems. In particular, examine the messages passed and data to be shared.

3.1.2 From the required message communication speed and pattern between subsystems, propose alternative architectures, bearing in mind the functional and non-functional requirements, desired software quality attributes, and constraints.

3.1.3 Evaluate the trade-off’s associated with each alternative. Select a suitable candidate architecture for the system. Document your selection criteria and rationale on your wiki.

3.1.4 Specify the selected candidate architecture using notation that describes the static programme structure, as well as, the dynamic programme behaviour. Use the notation discussed in the lectures.

3.2 *Design subsystem interface*

3.2.1 In the course of determining the system architecture, the responsibilities of the subsystem is clarified. The requirements initially allocated to the subsystem may have changed.

3.2.2 Re-visit your use case model to now add the relevant details.

3.2.3 Design your subsystem interface according to services it provides to, and requires from, other subsystems. It will be helpful to consider the messages passed to / from your subsystem to enact the key system usage scenarios.

3.2.4 Document your interface design by describing the Public Methods (i.e., functionality of the public API, parameters of the API, usage scenarios and design rationales). Your interface design serves as a contact, for other subsystems to proceed with design and implementation.

.

4. <u>**DELIVERABLES**</u>

- Candidate Architecture (per team)
- Rationale / Reasons for candidate architecture selection
- Subsystem Interface Design (per subsystem)