Assignments for Software Patterns (INMSP-08)

1. Pattern-based design (Group - 35%)

Due: December 6th

Design the architecture for a non-trivial system by selecting and applying patterns. Each group is free to select the system they want to design. It is advisable to use your current ideas and interests about a new (non-trivial) system you would like to work on – creativity and enthusiasm will of course be rewarded. You will come up with your own requirements. Your proposed system needs to be approved by your coach before the 1st review meeting. Consider the following guidelines:

- Use the PDAP method to design the architecture (*N. Harrison and P. Avgeriou, Pattern-Driven Architectural Partitioning - Balancing Functional and Non-functional Requirements*).
- Make sure you document the stakeholders, their concerns, the key drivers and the architecturally significant requirements.
- For every pattern you apply, record the architectural decisions (AD) you make. Document your decisions as described in the following paper: *Neil B. Harrison, Paris Avgeriou, Uwe Zdun. Using Patterns to Capture Architectural Decisions*. We expect you to document your decisions as in the template shown in Table 1 by including the following fields: Issue, Assumptions/constraints, Positions, Decision, Argument, Implications, Related decisions and Related requirements.
- The most appropriate patterns to use, are the architecture patterns as presented in the lecture or in *P. Avgeriou, U. Zdun. Architectural Patterns Revisited - A Pattern Language*. Of course you can also use patterns from other sources or the Internet.
- Make sure you integrate the patterns (or their variants) appropriately and come up with a coherent design.
- Each applied pattern should be accompanied with a reference to its source (e.g. citation to a book or a web-page)
- You do not have to fully document the architecture (see General Remarks)

2. Pattern-based recovery & evaluation (Group - 65%)

Due: January 31st

First part: Choose a non-trivial open-source system and recover its architecture by mining at least five patterns. Consider the following guidelines:

- Each group is free to select the OSS, but it must be approved by the coaches before the first review meeting to make sure it's a realistic case (a large and complex system whose patterns have not already been documented).
- Use the IDAPO method (see *K. Stol, P. Avgeriou, M. Ali Babar, Design and Evaluation of a Process for Identifying Architecture Patterns in Open Source Software*). Alternatively you can use another systematic approach (for identifying patterns) of your choice, after getting the approval of your coach.
- For inspiration you can consult The Architecture of Open Source Applications (http://www.aosabook.org/en/index.html)
- Make sure you recover also the stakeholders, their concerns and the key drivers. However, do not spend too much time on stakeholders and their concerns – your intuition should serve you well for the first iterations.
- The patterns you recover can be architectural, enterprise, design, analysis or other kinds, as presented in the lecture.
- Explain how exactly you found the patterns (e.g. they were explicit in the documentation, or they were implicit in the communication of the OSS community, or by looking at the code)
- Discuss the specific variants of the patterns applied.
- For every pattern you recover, record the corresponding architectural decision using the same template as in the first assignment.
- Each pattern recovered should be accompanied with a reference to its source (e.g. citation to a book or a web-page)
- Remember that patterns are not islands but closely interact: therefore do not just show individual patterns, but show the whole architecture as an integration of the patterns.
- You do not have to fully document the architecture (see General Remarks)

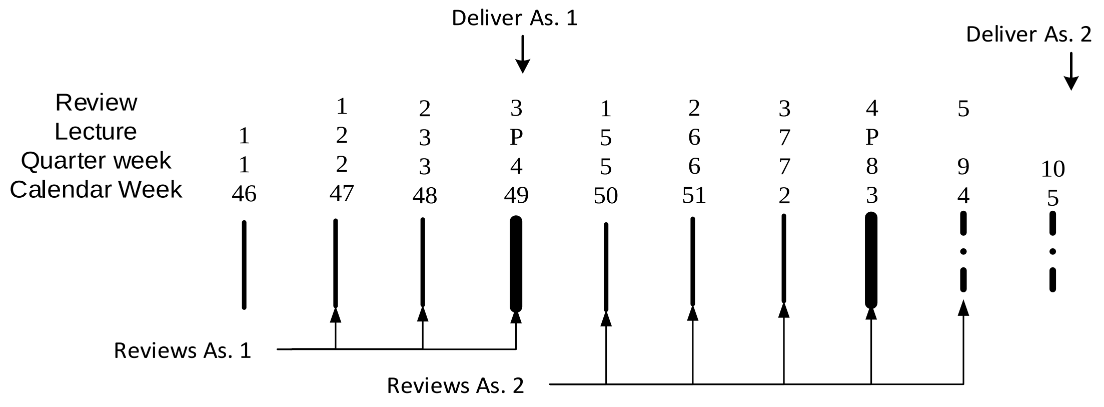
Second part: Perform an architecture evaluation of that system based on its patterns by considering the following guidelines:

- Use the PBAR method (see *N. Harrison, P. Avgeriou, Using Pattern-Based Architecture Reviews to Detect Quality Attribute Issues – an Exploratory Study*).
- Evaluate the consequences of the patterns on the system's quality attributes. In addition, to estimate the overall impact on quality attributes, take into account other factors (e.g. technologies, design decisions) that have direct influence on quality attributes.
- Propose ways to improve the system's quality attributes by incorporating new patterns or variants of existing patterns and show the re-engineered designs. Make sure you back up your proposed solutions with actual system design. Alternatively consider future functional and non-functional requirements and show how they can be accommodated through new patterns.
- The emphasis of this part lies on the proposed improvements based on patterns.
- You can use different parts of the Open-source system documentation to mine the desired information (e.g. discussion forum, code archive).
- You are encouraged to contact the corresponding open-source communities to validate the documented patterns and the improved design you have come up with.

A word of caution: Please make sure you work iteratively and incrementally with appropriate time-boxing in order to spend sufficient time in **both parts**. Note particularly that the second part accounts for 35% of the assignment’s grade (see grading template below) and you should therefore spend the corresponding effort.

General remarks

- Read the assignment descriptions thoroughly as they contain the answers to most of the initial questions you may have.
- Assignments 1 and 2 are made by **groups** of max. 5 students that are self-organized, i.e. you get to pick your own team-mates.
- Assignment **deliverables** should be uploaded on Nestor (file exchange of the group) each week by the corresponding deadline. All files delivered should be in PDF format. The uploaded files of the Assignments should be named as follows <GroupNumber>_<AssignmentNo>.pdf
- To support the coaches in providing optimal feedback in assignments 1 and 2, the **changes in the document** since the last version should be clearly visible. There are different options you can use to edit your document according to your experience and preferences: LaTeX + subversion, OpenOffice or MS Word, Google Docs etc.
- The **participation of all members** of the groups in the presentation of Assignments 1 and 2 in the classroom is mandatory. Exactly half the group must present the first assignment, while the other half must present the second assignment.
- For assignments 1 and 2 you can use the **documentation template** from the Software Architecture course (INMSA-08) but the architecture need not be fully documented: you are only expected to fill in the essential parts. The table at the end of this document highlights the relevant parts of the template for assignments 1 and 2 and the corresponding grades.
- The **time-plan** of the course including the lectures, reviews, deliverable chapters, presentations and delivery deadlines is illustrated in the following diagram.



Review		1	2	3	1	2	3	4	5	
Lecture	1	2	3	P	5	6	7	P		
Quarter week	1	2	3	4	5	6	7	8	9	10
Calendar Week	46	47	48	49	50	51	2	3	4	5

Template section	Assignment 1	Assignment 2
1. Business model: System context, Architectural relevant business parameters	Optional (extra points if provided)	5% - show system context and OSS community
2. Requirements: Stakeholders & concerns, Use case descriptions, Requirements, Risk analysis	10% - focus on stakeholders and concerns, key drivers, list of architecturally-significant FRs and NFRs	10% - focus on stakeholders, concerns and key drivers
3. Analysis: Architectural issues, Design alternatives	20% - follow PDAP to apply patterns as a list of design decisions and document them with the template	Not applicable
4. System architecture: Overview, Components and interfaces, Allocation to HW and SW	Only if the chosen system has a large HW part (e.g. embedded system). Then System and HW architecture sum up to 10% while Software Architecture 20%	
5. Hardware architecture: Reverse engineering, Fit in overall architecture		
6. Software architecture: Architectural views, Architectural patterns, Decomposition, Component responsibilities & interfaces	30% - show both the entire system as integration of patterns as well as the individual pattern details; show at most 2 views	30% - show the individual patterns, their variants and how they integrate in the entire architecture; show at most 2 views;
7. Architecture evaluation: Evaluation method (e.g. PBAR), Conclusions and Improvements	20% - show the benefits and liabilities of each pattern on the system as well as the eventual tradeoff due to the pattern integration. Simple pattern verification.	35% - follow PBAR to evaluate the quality attributes based on the patterns. Emphasize on recommendations on how to improve the system
Presentations: content and style, answering and asking questions	10%	10%
Process: Teamwork & meeting deadlines, Motivation, Initiative and Creativity, Following feedback	10%	10%