Theme 4.1 SE, SA Learning Task 1, M. Kindermans & M. van Mastrigt, group 10b.

Assignment 1

(a) How do the viewpoints as defined by Rozanski & Woods and the RUP 4+1 views as defined by Kruchten relate to each other?

In contrast to the 4+1 architectural view model by Kruchten, which uses 5 viewpoints, the model by Rozanski & Woods uses 7 viewpoints. Below are the viewpoints of Kruchten and how they relate to those of Rozanksi & Woods.

- **Logical view**: Like the *functional viewpoint* of Rozanski & Woods, it describes what functionalities the system offers.
- **Development view**: Like the *development viewpoint* of Rozanski & Woods, it describes those parts of the architecture that are involved in the development of the software.
- **Process view**: Like the *concurrency viewpoint* of Rozanski & Woods, it describes how concurrency is handled in the software. Besides the concurrency viewpoint, it is related with several other viewpoints like: the *information viewpoint* for the distribution, the *functional viewpoint* for performance and runtime behavior, but also the *context viewpoint* and *deployment viewpoint* for runtime dependecies and runtime performance. It is related to the *context viewpoint* in the way of explaining how system processes communicate.
- **Physical view**: This view depicts the system from a system engineer's point of view. So it is related to Rozanski & Woods' *deployment viewpoint*.
- **Scenarios**: This view is related to the following viewpoints: *context viewpoint*, *development viewpoint* and *deployment viewpoint*.

(b) The main difference between the definition of architecture as given by Rozanski & Woods and the definition of IEEE is the clause “the principles guiding its design and evolution”.

Why do you think the IEEE included this in the definition?

Because by including this in the definition, you are forcing the use of a certain view on the system and on the future. By doing this, everyone will not only have the same goal, but they are also trying to achieve this in the same way.

Give an example of a system and two principles “guiding its design and evolution”.

One of the principles of the definition of Rozanski & Woods is: "*It is not possible to capture the functional features and quality properties of a complex system in a single comprehensible model that is understandable by and of value to all stakeholders.*" You can see this in the many viewpoints they have defined.

(c) Chapter 5 of ISO/IEEE 42010 standard describes the contents of an architectural description. Compare this to the RUP SAD template. If there are any topics missing, is that a problem?

If missing topics are covered within other topics, it will not be a problem. But it looks like the stakeholders and the rationale behind certain choices will be missing. This is a problem, because it may be unclear why certain choices are made (from a stakeholder's perspective).

Assignment 2

Mobile Agents

(a) In the SAD Mobile Agents, paragraph 4.3, the choice for the agent framework is an important design decision. Write it down using the design decision format presented at the lectures.

Option Description:
The Java Agent Development Framework (JADE) is a middleware developed by TILAB for developing distributed multi-agent applications.

Pros:

- Meets the FIPA standards;
- Writting in Java, which makes it platform and hardware independant;
- Open-source software (LGPL license);
- Large community, which is helpful for answering questions and solving problems;
- Allows for running all the required agent options.

Cons:

The framework is written in Java, which could mean the software is slow.

Risks and Issues:

The last release was on 2 july 2009, while the document is written in june 2010. This could indicate slow development cycles, but could also mean the product is well matured.

Also little attention is given to communication between agent systems.

Assumptions and Constraints:

The framework is not developed as open-source, which might mean the development may stagnate.

Points of Note:

The Java VM can become slow if the codebase is too large.

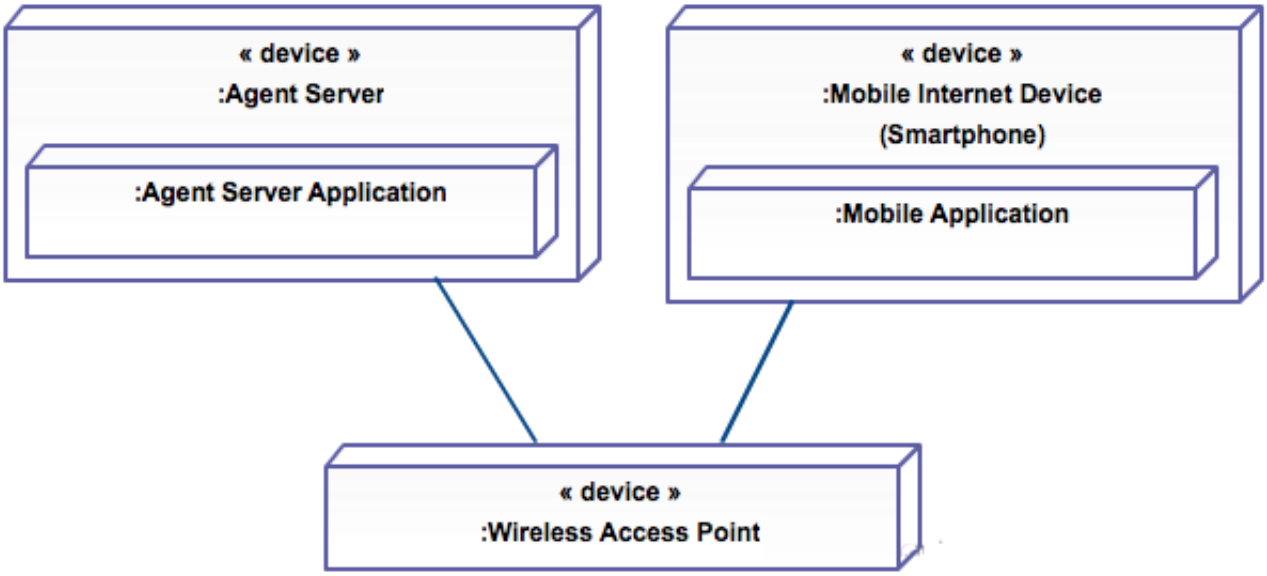
(b) Which RUP view is missing in this document? Is this a problem?

The *Process View* is missing which is a big problem. Since the JADE framework is used to meet the

FIPA standards, which allows for communication with other agent systems, it is essential that the way of communication is documented.

There is no information given on distribution, scalability, or performance. And no UML diagrams to represent process views, such as a Activity diagram to indicate the dependencies between agent systems.

(c) Give an UML deployment diagram to add to chapter 5.



GAIUS

(d) In GAIUS Architecture Description chapter 2 talks about "system roles". Are these comparable to stakeholders or to use case actors?

These are comparable to Use Case Actors.

(e) Explain the difference between stakeholders and use case actor. Give examples.

Stakeholders are people who have an interest in the system. This group of people may include investors, higher management, users of the system, or partners from system dependencies. An example is a company officer who invests in a system to improve their business processes, but who will never use actual the system himself.

Actors interact with the system. Actors are tighly coupled to the UML Use Cases. They might be humans sitting at the computer or other systems calling APIs or being called via APIs.

(f) To which RUP view does the information belong that is presented in 4.2.3?

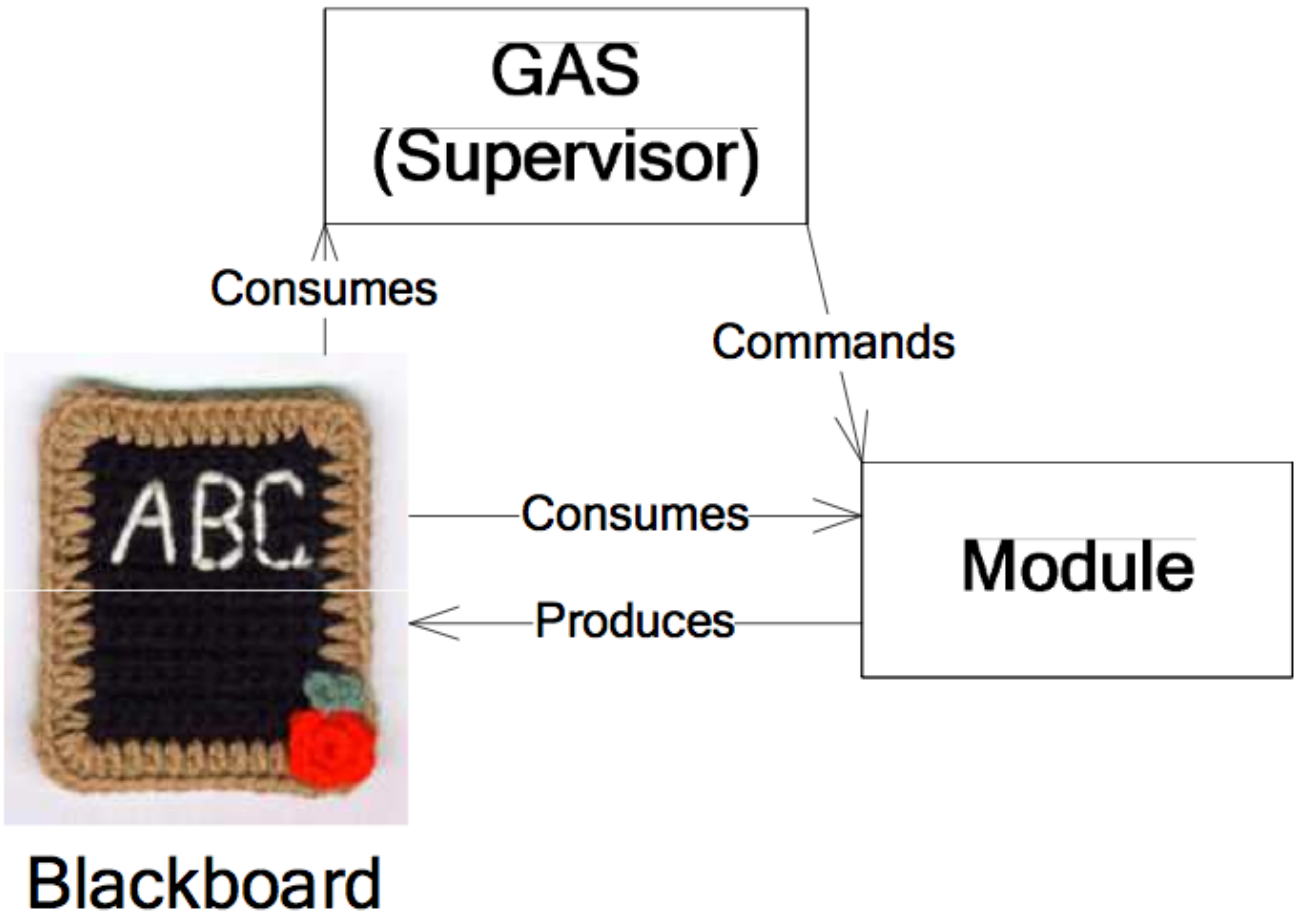
It clearly describes communication between components and therefore belongs to a logical view.

(g) In 4.3.1 the Blackboard pattern is mentioned. This is an example of an architectural pattern. Explain in a few sentences what an architectural pattern is.

Patterns are often defined as "strictly described and commonly available". A design pattern in general is a recognised and reused solution to a recurring problem in the field of software design. An architectural pattern is therefore a reusable solution to an architectural problem, which has a broader scope than the concept of a design pattern. These patterns are used in the field of software engineering and address issues such as (hardware) performance limitations, availability, and business risks.

Source: https://en.wikipedia.org/wiki/Architectural_pattern

(h) Explain the Blackboard pattern.



The Blackboard pattern is a behavorial design pattern. Which helps to reduce both complexity and duplication of code, as reduced coupling between sender and receiver - which improves the system's flexibility.

The Blackboard pattern in particular is used to coordinate separate systems that need to work together, or in sequence, continually prioritizing the actors. This is done through a shared storage, which acts as a messagebox that can be accessed by (physically) separate processes. A controller

monitors the values in the messagebox (or the properties on "the blackboard") and makes decisions about priorization of the processes (actors).

Source: http://social.technet.microsoft.com/wiki/contents/articles/13215.blackboard-design-pattern.aspx

Assignment 3

(a) What are the main quality attributes the stakeholders are interested in? Give three quality attributes for each stakeholder.

Development/Maintenance team:

1. Maintainability
2. Debugability
3. Testability

Management:

1. Configurability
2. Extensibility
3. Affordability

(b) Give two other possible stakeholders we could have considered for this system. What would their concerns be?

1. **End-Users**, their concerns would be:
    - Usability
    - Simplicity
    - Correctness
2. **Administrators**, their concerns would be:
    - Administrability
    - Usability
    - Autonomy

(c) Study the system (source code and available documentation). Write a SAD based on the template at www.rupopmaat.nl. Make sure to include UML component diagram(s), package diagram(s) and deployment diagram(s).

See attached `SAD Project DolVA.pdf` .

(d) Identify three important design decisions that have been taken for the system. Where these design decisions documented by the original developers? What were the alternatives? What decision would you have made, with what rationale?

1. **Design decision:** *API Based*

   Documented by original developers: Yes, it is documented. The given rationale is: "*Because of the simplicity of addressing there was made the decision to use a RESTful API. This reduces the complexity as much as possible.*"

   The alternatives: The alternative would be to let every endpoint communicate directly to the application layer. This would mean no intermediate API, which requires an interface for each individual endpoint.

   Our decision and rationale: We would have made the same decision, because there is less tight coupling between the graphical user interface of the CMS and the logical view of the CMS. This extra layer of abstraction makes communication more tranparent, better maintainable, and also makes debugging and testing easier.

2. **Design decision:** *Cordova App*

   Documented by original developers: Yes, it is documented. The given rationale is: "*Cordova makes it possible to develop without using the device's native APIs.*"

   The alternatives: The alternative would have been to build native apps in Java (for Android) and Swift (for iOS).

   Our decision and rationale: We would have made the same decision, because by using Cordova you combine best of both the browser possibilities and built-in native functionalities. Also Cordova makes developing an app for multiple platforms (cross-platform) a trivial task.

3. **Design decision:** *MySQL*

   Documented by original developers: Yes, it is documented. The given rationale is: "*Because there is a strong relational character between the data and there is a relatively small set of subjects.*"

   The alternatives: Yes, there were a lot of alternatives, e.g. Microsoft SQL Server, Oracle, PostgreSQL, MongoDB, and a lot of others.

   Our decision and rationale: We would have made the same decision, because of the wide use of MySQL and the strong relation between the different data.

(e) How could the architecture of the system be improved? Give suggestions for improvement, at least one for every view.

1. **Improvements to the logical view**

   Suggestion(s): There are no diagrams present at all. It would be better if there would also be made some static diagrams, like a component diagram or ERD.

2. **Improvements to the development view**

   Suggestion(s): A lot of artifacts of this view have been left out. It would be useful to add the file structure. This would make it easier to find certain files for new developers and it would improve the extendability. On the plus side open-source frameworks are used, which means documentation on file structure (etc.) can be found online.

3. **Improvements to the process view**

   Suggestion(s): It would be useful to add a desciption of how the scalability of the system has been improved. Also there are no diagrams to show the outline of the entire system and no documents explaining how the individual subsystems communicate with one another.

4. **Improvements to the physical view**

   Suggestion(s): There are only explanations why certain decisions have been made. It would be useful if there is some documentation available about which protocols are uses and how the hardware has been configured.

5. **Improvements to the scenarios / use case view**

   Suggestion(s): Only User Stories are made. Although they are very extensive, it would be better if there would also be made Use Case Diagrams and Use Cade Descriptions, because they give a clear overview.

(f) The standard views are probably too technical for Martin. Write a management view. You are allowed to make reasonable assumptions. (This exercise is about writing for a non-technical person and being able to imagine the things he's interested in.)

See attached `Management Review.pdf` .