Name: Jan Martijn van der Werf

Studentnr:

Software Architecture Exam – Possible answers

27 January 2015, 13:30 – 16:30

- This exam consists of 6 questions on 19 pages. Please check first whether you have properly obtained **all** pages.
- Enter the answers in the space allocated. In case you need more space, you can **use the back** of the pages. Make a proper reference to such an extra part on the back.
- When you have finished the exam, you should submit the complete package stapled in the correct order.
- Read carefully the question before you start answering!
- Reread your answers to check whether you really answered the question posed!

Question	Max. points	Awarded points
1	10	
2	15	
3	20	
4	20	
5	20	
6	15	
Total:	100	
Grade:	**Total / 10**	

This exam contains the answers on the questions.

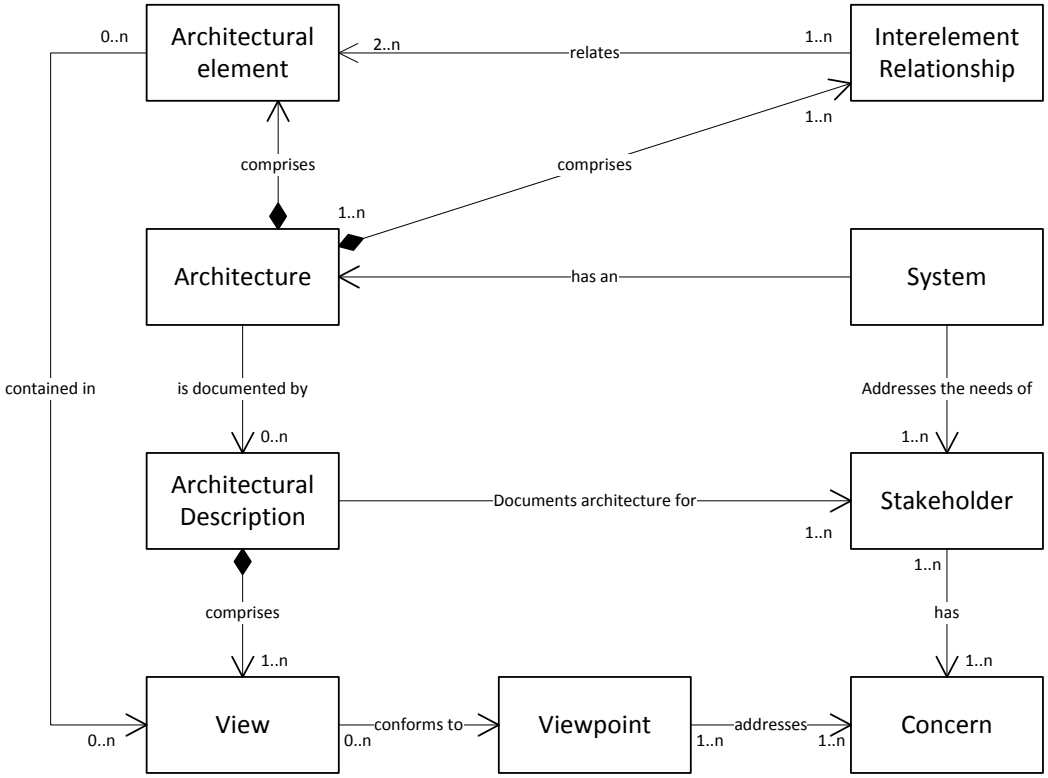
Note that for many questions, this is not THE answer, but a sufficient answer to gain all points.

Good luck with the exam!

1. Basic concepts (10p)

The definition of software architecture of a system used throughout the course is of Bass et al: “the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both”.

Consider the following model based on Rozanski & Woods explaining the relations between the different elements within the field of software architecture.



a. Give an example of how an architect can use Viewpoints to address Concerns of Stakeholders (1p)

The development viewpoint defines the guidelines that can be used to build developer views that show which code units are constructed.

Not mentioning both Viewpoint & view: max 0.5

b. Explain how an architect can use Viewpoints to address Concerns of Stakeholders (2p)

Viewpoints define patterns, guidelines best-practices and rules to build / define views for a set of concerns of a stakeholder

Only mentioning that viewpoints guide the structures used to address the concerns of a stakeholder: 1p

Only mentioning views / viewpoints: max 1p

c. Explain using the above model why an architect models Structures but documents Views. (3p)

Software is a large, complex system with many elements and relations between those. [OR A view is a representation of a set of structures]. So in a view not all elements and relations are depicted (as indicated in the model). (1p) By using views, the architect can organize and group the different elements and relations (1p) to address Concerns of Stakeholders (1p)

Another view on architecture is by Taylor et al. in their book "Software Architecture: Foundations, Theory, and Practice": "A software system's architecture is the set of principal design decisions about the system".

d. Explain how this definition complements the definition used throughout the course (Bass et al.) (4p)

Considering the software architecture as a set of structures of elements and their relationships and properties, is the end product (1p) of a whole chain of design decisions made by the architect (1p). So the definition of Taylor et al focus more on the process of creating an architecture (1p), whereas Bass et al focus on the artifact itself (1p).

Design Decisions do not equal architectural significant requirements, as requirements do not carry any decision. (-1p)4

2. Software product lines (15p)

Consider the definition of a Product Line: “A product line of software is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way”.

a. Explain “sharing a common, managed set of features”. What means common? What means managed? (3p)

As within a software product line many systems have similar features, a software product line is architected in such a way that this common set of features is designed to be used within all members of the software product line. So, common stands for the part that they are reused within the family, and managed means that the core containing this common set of features is explicitly architected for reuse, easing maintainability, the creation of new members, etc.

b. Explain: “common set of core assets in a prescribed way”. What are core assets? What means prescribed way? (3p)

The core assets is the set of assets and software elements that together populate the architecture of the software product line. Assets include designs, user manuals, test cases, etc. These assets are architected and built such that they support multiple members within the software product line, so that reusing is much simpler. Prescribed in this context means that the assets are specifically designed for the purpose of reuse, with variability points, and should be used by the members of the software product line.

c. Give an example of a well-known product line. (1p)

Call of Duty,

d. Explain why your example in question c. is a product line. (2p)

It is a game with many versions, that all depend on a common set of features, with built-in variation points. For example, there are variation points for the different platform on which the game operates. But the different versions also share common assets for e.g. graphics and the game engine.

e. Explain the notion of a Variation Mechanism in product lines. (3p)

As in a Software product line, one does not want to change code in order to support different variations, the software element should already have built-in ways of handling variations. The way this is taken care of, is by introducing a variation mechanism that allows for the selection of different versions of elements that have the same interface but different behavioral or quality attribute characteristics.

f. Explain how plug-ins and add-ons realize a variation mechanism in a product line. (3p)

Plug-ins and add-ons make use of a predefined interface of the software product to add at runtime new functionality to the system, without having to change the core product. In this way, they add new behavior or modify existing behavior. By changing a plugin or add-on we thus create a new member of the software product line. Add-ons and plugins are a realization of the “Selection of different versions of elements that have the same interface but different behavioral or quality attribute characteristic” variation mechanism.

Not mentioning the interface / dynamic: -1p

3. Patterns and quality attributes (20p)

An architectural pattern establishes a relationship between a context, a problem and a solution [1].

a. Explain how an architectural pattern establishes this relationship. (3p)

A pattern describes a successful architectural solution to a problem that arises in a recurring, common situation.

Only stating that a pattern is built by combining tactics, is not sufficient, as a tactic is often required to adapt a pattern to a specific situation: -1p

Not stating it is a common / recurring context / problem: -1p

b. Explain the relation between a viewpoint and an architectural pattern. (3p)

A viewpoint combines patterns (1p) to address a selected set of concerns of a stakeholder (1p). Thus, a viewpoint helps in guiding the selection of which patterns to use in a given situation (1p)

Given is the following summary and evaluation of the Broker pattern

> **Summary**:
> Providers register their function to a Broker. When a Client requires some functionality, it sends its request to the broker. The Broker decides which Provider to contact, sends the request, and forwards the response of the Provider to the Client.
>
> **When to use it**:
> Unknown number of providers that offer certain functionality
>
> **When to avoid it**:
> Requests or responses are too large to be routed via a Broker

c. Explain the Broker-pattern in terms of context, problem and solution. (3p)

Context:

A situation in which (distributed) partners need functionality offered by other partners, but partners do not know which partner to use.

Problem:

How to use functionality if partners do not know each other?

How to provide dynamic binding?

Solution:

Elements are divided in broker, clients and providers. Clients and providers only communicate via the Broker

Another example of an architectural pattern is the SOA Registry.

Summary:
Providers register their function to a Register. When a Client requires some functionality, the Register returns an appropriate Provider. The Client then contacts the Provider to handle its request.

When to use it:
Requests or responses are too large to be routed via a Broker

When to avoid it:
Providers not always responsive

d. Explain the Registry pattern in terms of context, problem and solution. (3p)

Context:

Consumers need to be able to understand and use providers without any detailed knowledge of their implementation

Problem:

How to support interoperability of distributed components?

Solution:

Elements are divided in Registry, Client, and Provider. Providers register at Registry. If a client requires functionality, it asks for a Provider at the Registry. Client then directly communicates with Provider.

A solution for a pattern is determined and described by Bass et al.:

- A set of element types
- A set of interaction mechanisms or connectors
- A topological layout of the components
- A set of semantic constraints covering topology, element behavior and interaction mechanisms.

e. Describe the role and use of each of these four aspects for the solution of the Register-pattern. (4p)

1 point per element, ½ if at least half of the elements are present, otherwise 0p.

Only describing the role, without mentioning the Registry: at most 1 ½ p (½ per element)

A set of element types:

Client: module / component that requires some functionality

Provider: module / component that delivers some functionality

Registry: module / component that maintains a list of which provider offers which functionality and where providers can register themselves.

No explanation? At most ½p

A set of interaction mechanisms or connectors:

SOAP or REST connector

(A)synchronous Message Channel

A topological layout of the components:

See picture on slides. OR description of that picture

A set of semantic constraints covering topology, element behavior and interaction mechanisms:

Providers can register themselves at the Registry

Clients can request at the Registry which Providers offer which functionality

Client can approach a Provider with a request for functionality

Provider can return a result upon a request of a Client

Within a public-transport organization, the software architect is designing a payment system that will be used in all buses within the region. The region the organization operates in is a rural area with a few villages and a large city. Within the city, the WiFi network coverage is good, however between the villages, there is no WiFi connection, and the GSM network stutters from time to time. The main quality attribute the architect is concerned with is availability.

Based on the current requirements, the architect is considering whether to use the Broker or the Registry pattern.

f. Explain **and show** how the architect can use tactics to decide which of the two patterns is best applicable in his situation. (4p)

Only giving a generic answer: at most 2p.

Generic: *Patterns are often too generic for the specific situation, and therefore need to be adapted to the situation. Tactics help the architect in deciding how to adapt the pattern. Applying a tactic results in a solution that better satisfies the intended quality attributes.*

In this situation, the architect has to consider availability. The main problem is to decide which intermediate to choose, the Broker or Registry.

Sketch of the architecture: Bus contains payment terminals (clients) that need to connect with the payment service via some networks (WiFi or GSM). So what is the provider? The payment service or the network connections?

Tactic 1: Fault detection, use connections as hot spares.

> As the network is unreliable, a backup is needed: if one does not work, present the client the other connection.

This still makes no appropriate choice between the two patterns.

Tactic 2: instantly swap type of connection (GSM / WiFI) on fault

> As the connection might drop during communication, we need to be able to swap instantly. Then all communication between client and provider should be handled via an intermediate, so that if the connection drops, the intermediate can directly choose the other connection

Following these tactics, the Broker pattern is best applicable in this situation.

4. Quality attributes (20p)

a. Functional requirements and quality attributes go hand-in-hand. Explain the relationship between functional requirements and quality attributes. (2p)

Quality attributes annotate functional requirements.

Only mentioning they are orthogonal: 1p

b. Explain how quality attributes are addressed in the model presented in **question 1**. (3p)

Quality attributes are indirectly addressed by the concerns of the different stakeholders. As such, they appear in elements on different views. Each quality attribute thus creates a perspective on the views within the architecture description.

Only mentioning that concern equals quality attribute: at most 2p

c. Tactics can be used to achieve required quality attributes. Explain the relationship between tactics and the quality attribute scenario. (3p).

Tactics are a single, simple solution to improve the satisfaction of a quality attribute. The quality attribute scenario depicts how a quality attribute is threatened and how this can be measured. Tactics are solutions to solve these threats.

To analyze whether a software architecture satisfies the specified quality attributes, one can:

- Model the architecture to enable Quality Attribute Analysis
- Use Quality Attribute Checklists
- Perform thought experiments and Back-of-the-envelope analysis
- Create prototypes

d. Explain how each of the approaches can be used to assess the quality attribute, and give for each an example. (5p)

Examples forgotten? Max -2p

Many examples are possible!

Model the architecture to enable Quality Attribute Analysis:

Make an analysis model of the quality attribute to use analysis techniques to verify whether the architecture satisfies the quality attributes

Example: Queueing model of a MVC-pattern / Petri nets for analyzing concurrency properties

Use Quality Attribute Checklists:

Standardized checklists to ensure that each aspect of the quality attribute is addressed in the architecture.

Example: Security checklist containing elements like 'SQL-injection', `XSS-prevention', etc.

Perform thought experiments and Back-of-the-envelope analysis:

Make quick sketches and walk through the architecture whether it satisfies the quality attributes

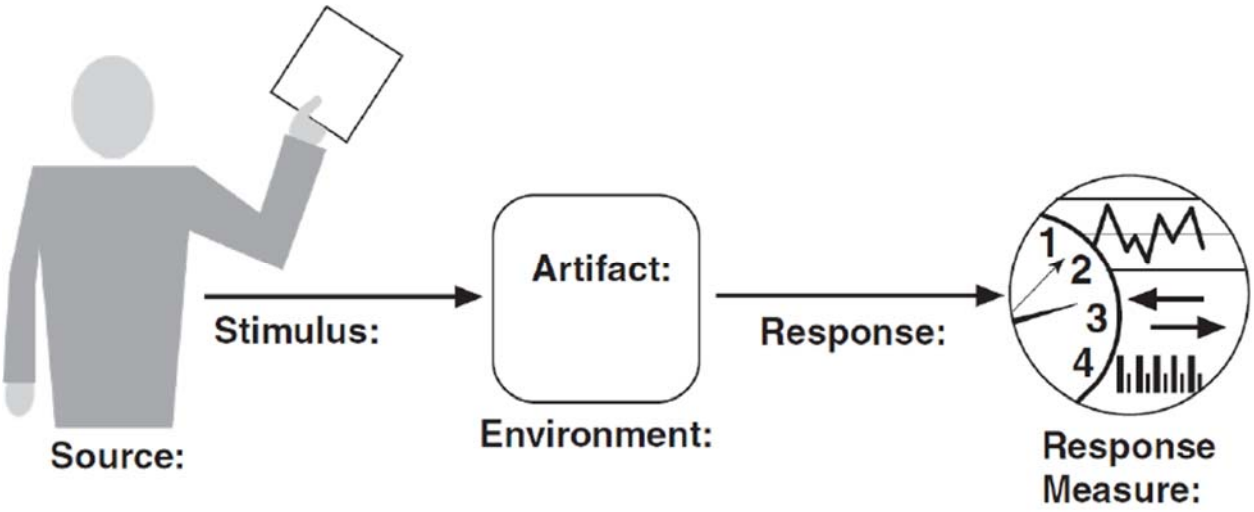
Example: Walk through of use cases to test whether the system responds as intended

Create prototypes:

Test or determine behavior of certain features or functionality by creating a proof-of-concept e.g., to see whether frameworks can be used

Example: Mockups / Rapid Prototyping tools.

e. An Architectural Significant Requirement in the public transport system as described in **question 3** is availability of the Broker. Create a concrete scenario for this quality attribute in the generic scenario depicted below (3p).



Source: user / client with travel card

Stimulus: card provided to card reader

Artifact: Broker

Environment: Rural area with limited GSM-network traffic

Response: Bus trip registered by system

Response Measure:

- amount of trials to register travel
- Eventually user is registered
- User / client gets registered message within 1s.

f. Explain your scenario. (4p)

Explanation how source – stimulus – response is measured. (2p)

Explanation how this links to availability (2p)

5. Functional architecture (20p)

A functional architecture model is an architectural model from a usage perspective.

a. Is the functional architecture model an informal, semi-formal or a formal notation? Explain your answer. (3p)

Formal ➔ 0p

Not Formal: no mathematical reasoning possible

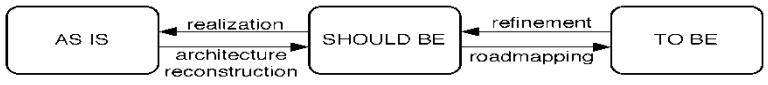
Informal ➔ max 2p

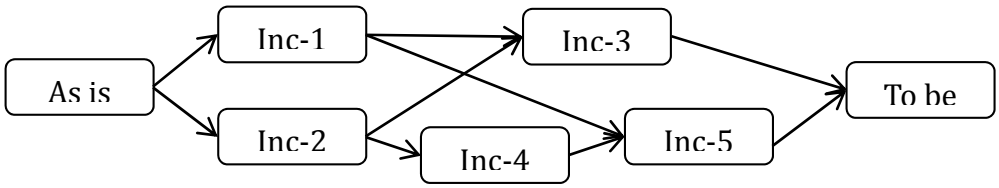
Not Informal: there are no tools that produce true FAMs, there are guidelines to notation but very loosely. However, some rudimentary analysis is possible and therefore informal is not the best option for this question .

Semiformal ➔1p + explanation max 2p

Functional architecture Model is a semiformal notation. It has guidelines on how to draw modules, the elements have a semantics, what flows there are between modules / features and their semantics. Rudimentary analysis is possible.

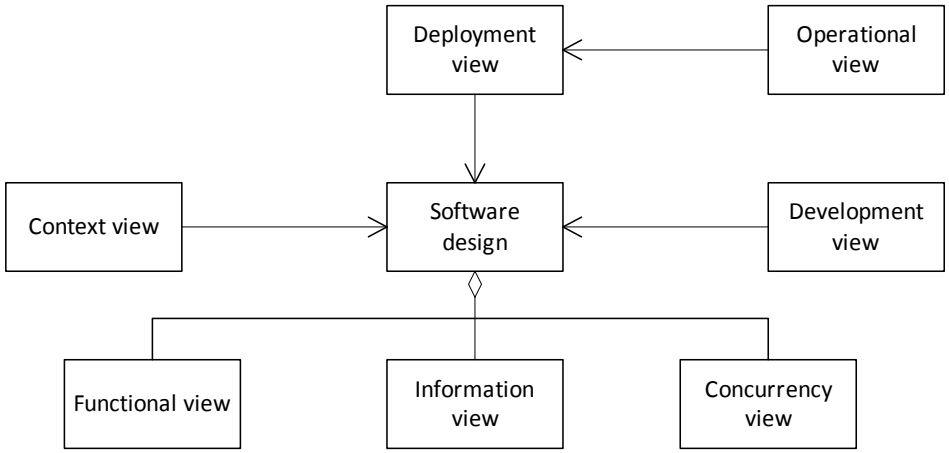
b. One of the uses of a functional architecture model is roadmapping. Explain how the functional architecture can be used for roadmapping. (3p)





Functional architecture can be used to describe a to-be situation. It depicts the final set of modules and their interaction. Based on the interactions between modules, one can develop a strategy to incrementally build a system, taking care that modules that are realized in earlier stages already take future realizations into account.

The viewpoints of Rozanski & Woods in "Software Systems Architecture" are catalogued in 7 viewpoints, as shown in the figure below.



c. Explain how the Functional Architecture can be used in the Context view. (3p)

The FA depicts which external modules and features are required for functioning of the system under architecture. As these external modules are not part of the system, they belong to the context of the system. Thus displaying the System under Architecture as a black box with information flows to its external modules defines the Context from a functional perspective, and therefore, this is a Context view.

Only stating that stakeholders know which functionality goes where in the system: 1p

d. Explain how the Functional Architecture can be used in the Information view (3p).

The functional architecture depicts which modules have which features, and how these features can interact via information flows. Scenarios, defining the order in which these features can interact and the objects that are communicated define an information flow through the system, which is an Information View.

Only mentioning that vertical axis of model is operational – tactic – strategic and horizontal axis is flow input – processing – output is not sufficient. This is a guideline on how to draw the model, and not on how information flows through the system… 1p.

The dynamic structure of the software is an important aspect to be taken into account. To model dynamic structures, we use Component-and-Connector structures.

e. What are the elements of a Component-and-Connector structure? Explain them and give an example for each. (2p)

A set of elements that have runtime behavior (components) (1/2 p) and their interactions (connectors) (1/2p), e.g. Clients that use functionality from a server (1/2p) via a Call-Return interaction (1/2p), like the RPC pattern.

Not mentioning runtime / behavior / interaction / communication: 0p.

Only mentioning interaction / communication between components: max 1p

f. Explain how the Component-and-Connector structure is embedded in the functional architecture model. (2p)

The modules depict which features are called by other modules via the information flow (1p). Each information flow is an interaction (connector) (1p).

Alternative: Scenarios depicting runtime behavior of the system

We distinguish two types of dynamic structures: horizontal and vertical behavior.

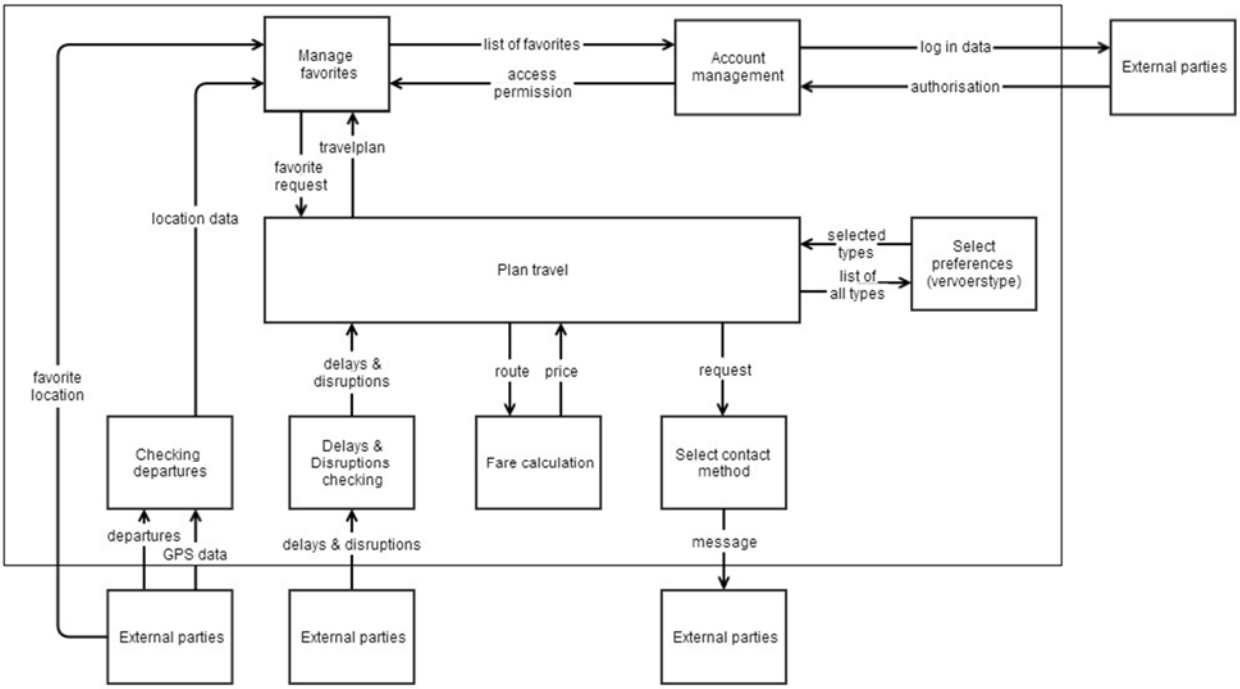
g. Explain horizontal and vertical behavior. (2p)

Horizontal behavior: The flow of information / data over the different modules in the system

Vertical behavior: The dependencies between and the ordering of features within a single module.

Swapping Hor. And Vert. behavior: max 1p

An example of a Functional Architecture Diagram within the domain of the Public Transport organization is depicted below. It describes how the "Plan travel" module uses functionality of other modules to derive a list of possible routes and the associated costs



h. Give an example of horizontal dynamic behavior using the figure above. (1p)

If consequent with explanation of previous page: max ½ p

i. Give an example of vertical behavior for this system. (1p)

Plan travel module: information flow route should always precede information flow price.

If consequent with explanation of previous page: max ½ p

6. Architecture Implementation & Evaluation (15p)

a. Architectural erosion is an imminent danger for any software product. In other words, the implementation of an architecture will almost certainly drift away from the intended architecture. Explain why. (3p)

Mapping of the architecture upon the realization is difficult to automate, and therefore a human effort. As the software architecture focuses on the main design decisions, many decisions required for the realization are not filled in, or are not foreseen at all. There are mainly guidelines on how to address such issues. The development team consists of different people working together to realize the software. As these people differ, they will make different decisions when the architecture has not filled in that decision.

b. Give two examples of how the risk of architectural erosion can be minimized. (3p)

Have guidelines on how to retrieve, make and document realization decisions

Use code templates

Use tools to enforce architectural constraints

Mark documentation out-of-date when erosion occurs

Schedule documentation/code synchronization times

To assess the quality of a software architecture, one can use different tools and techniques.

c. Explain the activities of Architecture Analysis and Architecture Evaluation and what their difference is. (3p)

Architecture Analysis is the activity of verifying whether the architecture satisfies a set of properties, i.e., whether the architecture is correct with respect to these properties. Architecture evaluation focuses on validating /ensuring that it is the correct architecture. So, evaluation is ensuring that it does what it is supposed to do, whereas analysis focuses on whether it does it correctly.

An important tool to evaluate a software architecture is the Architecture Trade-off Analysis Method (ATAM).

d. Explain how ATAM helps in improving the quality of a software architecture (3p)

ATAM gathers all important stakeholders together to discuss the different Quality Attributes a software system needs to satisfy. In this way, the architect can evaluate whether he targeted the right Architectural Significant Requirements, and whether the proposed architecture indeed satisfies all criteria. Further, the risks, weaknesses and trade-offs of the architecture are made explicit. Based on these outputs, the architect can test whether the architecture he created addresses the concerns of the stakeholders appropriately, and update it when necessary.

e. An important aspect of the ATAM are the intangible outputs of the method. What are intangible outputs, and why are these important? (3p)

Examples of intangible outputs are:

- Sense of community on the part of stakeholders
- Open communication channels between architect and stakeholders
- Better overall understanding of architecture, its strengths and its weaknesses
- …

These intangible outputs are important to create awareness of why the architecture is as it is: why certain decisions or tradeoffs are made, and to get people involved and dedicated to the architecture and system. (Hopefully it also helps in minimizing architectural erosion)

End-of-Exam