NTNU
Norwegian University of Science and Technology

Faculty of Informatics, Mathematics and Electronics

Department of Computer and Information Sciences

ENGLISH

Examination results will be announced: 16. June

Exam in the subject
TDT4240 Software Architecture

Monday 26. May 2008
9:00 am – 1:00 pm

Aids code C:

Simple calculator allowed.
These specified printed documents are allowed:

- IEEE (2000), "IEEE Recommended Practice for Architectural Description of Software-Intensive Systems", Software Engineering Standards Committee of the IEEE Computer Society.
- Kruchten, P. (1995), "The 4+1 View Model of Architecture", IEEE Software, 12(6).
- English-Norwegian dictionary (or to your native language if your not Norwegian) and/or a English thesaurus (English-English).

Contact person during the exam:

PhD-student Odd Petter N. Slyngstad, phone 7359 4483, mobile phone: 930 48 411

The points show how much each problem is worth in this exam. For each problem, each question has the same weight unless otherwise stated. The exam has 6 problems giving a total of 70 points. The remaining 30 points are credits awarded from the software architecture project.

Good Luck!

Salah Uddin Ahmed and Bian Wu

Controlled 8th of May 2008

Problem 1: Various questions (15 points)

Answer these questions in short:

1.1 What is Bass, Clements and Kazman's definition of Software Architecture (the definition in the textbook)?

Def: A software architecture is the structure or structures of a system consisting of software components, their external visible properties and the relationship between them.

1.2 What is the purpose of the CBAM architecture evaluation?

The purpose of the CBAM is to evaluate what tactics give most benefit from the given cost (find the tactics with highest return-on-investment.

1.3 What is an idiom (related to design pattern)?

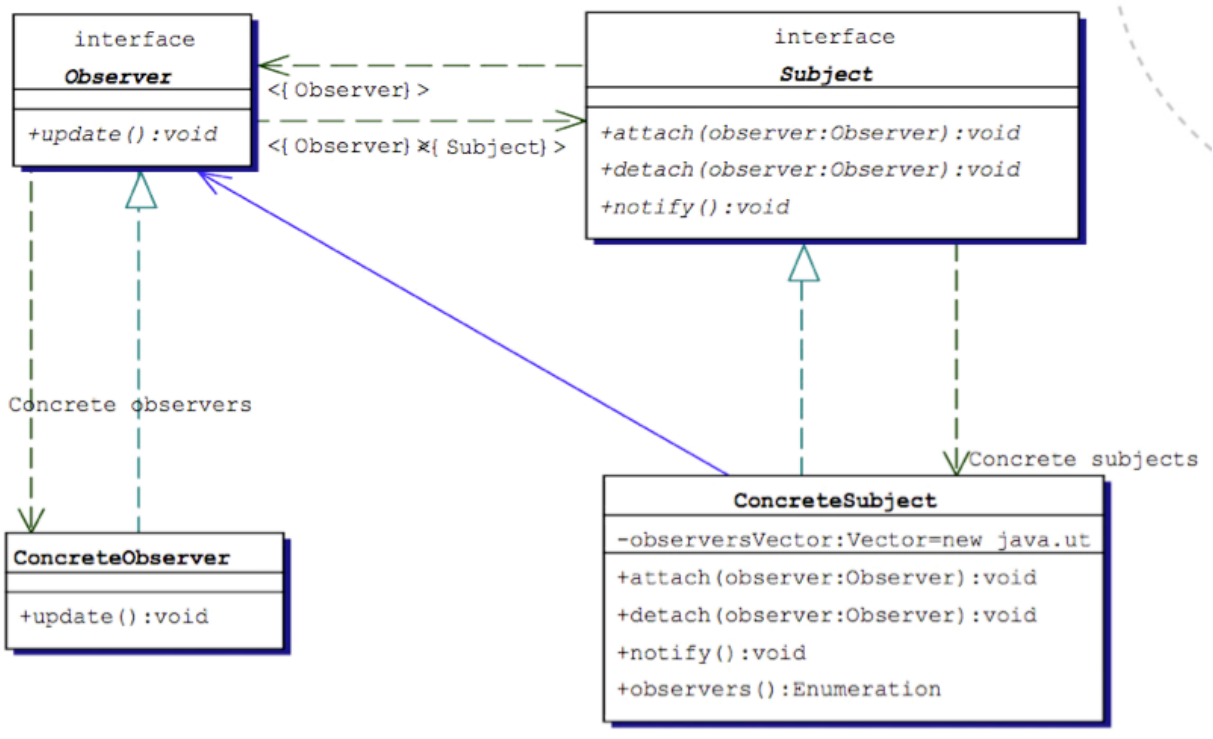
An idiom is a low-level reusable pattern specific for a particular programming language.

1.4 What is architectural drift?

Architectural drift is changes of architecture resulting in lack of coherence and clarity.

1.5 Describe the Observer design pattern and advantages of using it.

The Observer pattern is used where you need to update several objects when one object is changed. This pattern is used to solve issues related to high coupling.



1.6 Name five outputs from ATAM.

Outputs from ATAM: A utility tree, a set of sensitivity points and tradeoff points, a set of risks and non-risks, a set of risk themes, mapping architectural decisions (tactics) to quality requirements, concise presentation of the architecture, and articulation of the business goals.

1.7 What is Attribute-Driven Design (ADD) according to the textbook (describe)?

ADD is an approach to defining a software architecture that bases the decomposition process on the quality attributes the software has to fulfill. It is a recursive decomposition process where, at each stage, tactics and architectural patterns are chosen to satisfy a set of quality scenarios and then functionality is allocated to instantiate the module types provided by the pattern. ADD consists of these steps: 1) Choose module to decompose, 2) Refine the module according to a) Choose architectural drivers, b) Choose architectural patterns that satisfies architectural drivers, c) Instantiate modules and allocate functionalities, d) Define interfaces of the child modules and e) Verify and refine use-cases and quality scenarios.

1.8 Name the three main areas of performance tactics described in the textbook.

Resource demand, Resource management and Resource arbitration.

1.9 Explain how usability can be related to software architecture.

There are some usability issues that are related to software architecture and the structure of the system due to decisions that affects the whole system, e.g. Undo, and Re-do functionality. There are underlying features of the system that affects large parts or structures of the system. Issues related to look-and-feel of the system is not related to the software architecture.

1.10 How can issues of the main stakeholders of a system be addressed in a software architecture documentation?

The different main stakeholders can be addressed by providing different architectural views that allows different stakeholders to focus on different parts of the system, e.g. process view for system integrators, development view for project manager etc.

1.11 What is a reference architecture?

Reference architecture is a general architecture that can be used for solving the same problem for various of customers/needs. It defines standard decomposition of system components and standard decomposition of functionality of a general problem.

1.12 Explain how the architectural choices to obtain modifiability and performance in a system affect each other.

If modifiability has the main focus in a software architecture, it can lead to inefficient performance due to usage of many layers, and/or that the system has to go through many components to process or get information.

1.13 What is an architectural strategy according to the textbook?

An architectural strategy is a collection of architectural tactics (architectural decisions).

1.14 What is the purpose of the IEEE1471?

The purpose of IEEE1471 is to describe what is required to produce a sufficient software architectural documentation and it describes all the elements that are required.

1.15 Why can it be necessary to reconstruct a software architecture?

The software architecture might never have been documented, the architecture might be outdated, the architecture might be lost, key persons of a system might have left the company, etc.

Problem 2: Service Oriented Architecture (SOA) (10 points)

2.1 Service Identification Framework (SIF) (5 points)

Illustrate, describe and explain the *Service Identification Framework* (SIF) and its main parts. Also describe the requirements that must be taken into account when the SIF is used.

The SIF consists of 1) Service identification (identifying the required services needed in the system), 2) Service definition (defining the services and its interfaces) and 3) Service implementation (the actual implementation of the services). SIF needs to take both business requirements (architecture, organisation, rules etc.) and technical requirements into account (architecture, infrastructure etc.).

2.2 Service components in SOA (5 points)

Illustrate and describe the *Service Component and its parts* as given in the Service Component Architecture.

The service component consists of an implementation, an interface and reference. The interface typically consists of a Java interface and a WSDL interface and the reference does the same. The implementation can consist of several things like business process, Java code, adapter, state machine, business rules, human task, selector and mediation.

Problem 3: Creating a game architecture (5 points)

Describe and explain the process of creating a game architecture according to the book "Game Architecture and Design" by Rollings & Morris.

The process described in the book consists of three steps: 1) Find tokens (find any thing related to the gameplay in the game both playable things, game environment, score etc.), 2) Analyse interaction and events (create a token interaction matrix where you look at the interaction between all the tokens in the game, trace the events, and create a finite state diagram for NPCs), 3) Create logical view using tokens (tokens and token interaction can be used to sketch gameplay logical view and other views as well).

Problem 4: ATAM (5 points)

Do the step 6 (Analyze the architectural approaches) in the ATAM process based on the following information about a system for selling tickets over the web:

Utility tree:

- *Availability:*
  - *Scenario A1. The system should have less than 1 minute of downtime in a week. (M,H).*
- *Security:*
  - *Scenario S1: 99.9% of all money transactions should be safe. (H,M).*

Identified architectural tactics:

- *Authenticate users* – use username and password to provide authentication.
- *Authorize users* – use access control patterns to give the approved rights to data or services to the authorized users.
- *Maintain data confidentiality* – apply encryption to data and communication links to protect data from unauthorized access.

Scenario A1 is not relevant.
Scenario: S1: 99.9% of all money transactions should be safe (H,M)
Attribute: Security
Environment: Normal operation
Stimulus: Money transaction
Response: 99.9% transactions are safe

Tactic	Sensitivity	Tradeoff	Risk	Nonrisk
T1.Authenticate users		T1		N1
T2.Authorize users		T2	R1	
T3.Maintain data confidentiality		T3	R2	

Reasoning
The tactics used are well-known and well-proven tactics for security. T1 is considered safe and easy to implement. T2 and T3 should be specified more in detail to know for sure that the provided the security required. T2 and T3 is very dependent on the choice of algorithm and design.

Sensitivity points/tradeoff points:
T1: Tradeoff between security and usability
T2: Tradeoff between security and usability
T3: Tradeoff between security and performance

Risk/Nonrisks:
N1: T1 should be considered a safe tactic with minimum side effects.
R1: T2 could be difficult to implement and it is very important that the right approach and design is chosen.
R2: T3 is very dependent on the algorithm used in terms of how secure the system will be and the performance of the system.

Risk themes:
The whole security of the system depends very much on the actual implementation of the tactics (especially T2 and T3).

Problem 5: Choose the correct architectural pattern (5 points)

Peter N. Erd is hired as a software architect in the Big Mess software project, where his job is to choose the correct architectural pattern for the system. The system should have the following qualities:

- It should be easy to replace components of the system also in run-time
- To ensure consistency of the system, all information should be maintained in one place
- If the data fails, the system should go back or stay in a consistent state

Help Peter to choose among the following architectural patterns and **motivate** your choice:

1. Pipe and filter
2. Layered
3. Blackboard
4. Task Control
5. NASREM

The architectural pattern that fits best to the description is the **blackboard pattern**. The blackboard pattern has a very loose coupling to its components, which enables run-time replacements and configuration of components. The central part of the blackboard pattern is a central database with transaction support, which will take care of the consistency and data failure issues mentioned above.

Problem 6 Create an architecture (30 points)

Read the description below and do the following:
6.1 Identify the ***architectural drivers*** for the system described below (5 points)

The architectural drivers of the system are that each part of the system can be replaces (modifiability) and that the interfaces should be kept the same between the different parts of the system even if the system is changed (modifiability). In addition, performance will be very important to this system, as there are a lot of computations going on.

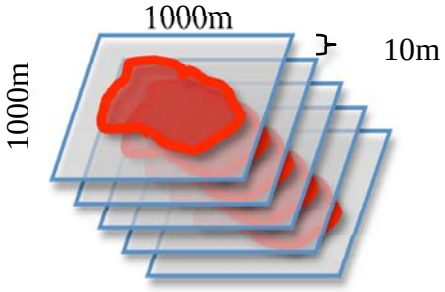
6.2 Choose and describe suitable ***design and/or architectural patterns*** for the problem described below, and describe how the patterns affect the quality attributes (5 points)

The most obvious architectural patterns are model-view-controller pattern to make it easier to change the GUI and pipe-and-filter pattern to describe the filtering of data going on. Various design patterns can be used, but one candidate is the observer-pattern to implement the model-view-controller.

6.3 Create ***architecture views*** of the system described below. The architecture must be described in two views according to the 4+1 view model: *Logical and Development view* (20 points)

Motivate for your choice of quality attributes, architectural drivers, design patterns and the architectural patterns used in your architecture. State your assumptions.

Software for an Oil Reservoir Computation System



The software described here is a system for performing computations related to oil reservoirs (areas in the ground where the oil is stored). The data input of the system is a large set of horizontal 2D-scans (2d-pictures) of the oil reservoir covering an area of 1km x 1km. A horizontal 2D-scan is produced for every 10m making it possible to make a 3D-representation of the reservoir.

Functional requirements:

- F1. Improve the horizontal 2D-scans by removing noise in the picture.
- F2. Improve the horizontal 2D-scans by sharpening the edges in the picture. This operation must be carried out after the noise has been removed.
- F3. Transform the horizontal 2D-scans to black & white picture, where the black area indicates oil and the white area indicates earth, rock or similar. The transformation is based on several parameters that maps colour values to what should be recognized as oil. This transformation must take place after noise removal.
- F4. Merge the pixels in the black and white horizontal 2D-scans that are close together to make solid graphical 2D-objects. This functionality is aimed at recognizing lungs of oil in the ground and removing additional noise in the picture. The result of the merge process is a picture that only consists of solid 2D-objects.
- F5. Transform a set of horizontal 2D-scans to a 3D-model representation. This operation will also take some parameters into account that will decide how the 3D-

objects will be represented and how many 2D-scans should be input to the 3D-model.

- F6. Compute the total volume of oil from 3D-models.

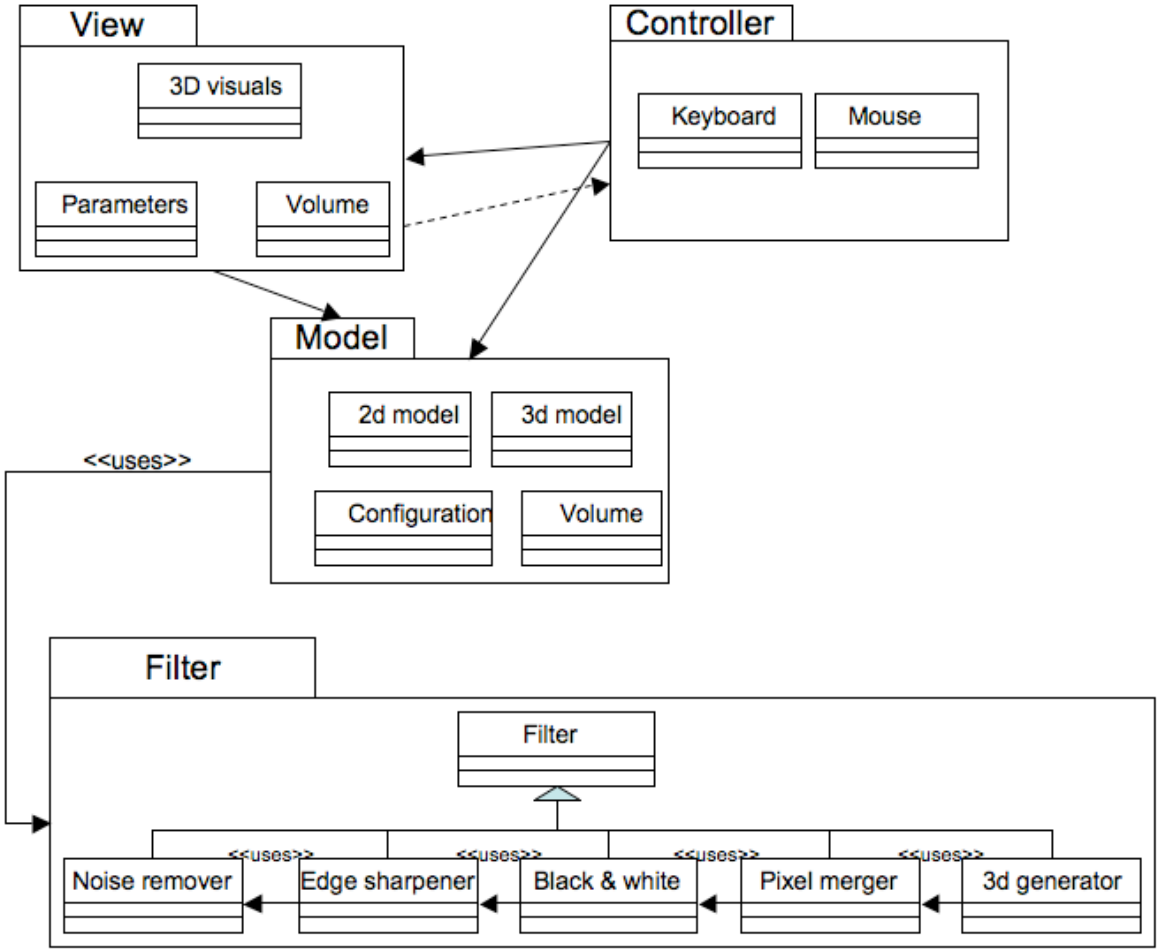
User interface requirements:

- U1. The user interface must provide text boxes for the input parameters for the system that will change how the process is performed.
- U2. The user interface must show a 3D-visualisation of the oil reservoir.
- U3. The user interface must display the volume of oil found in the oil reservoir.

Quality requirements:

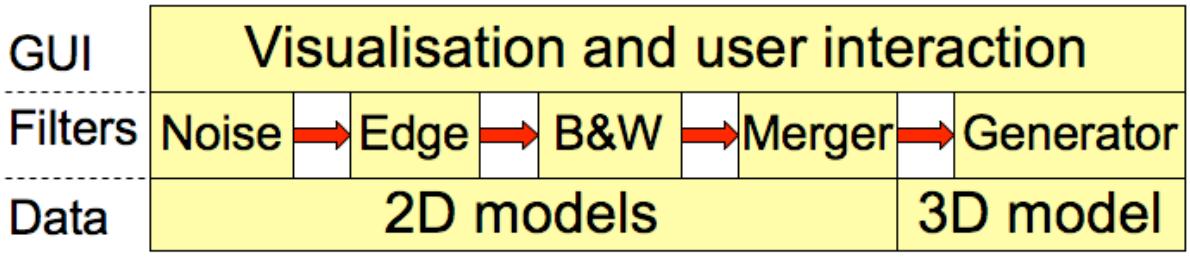
- M1. Modifiability: The various parts in the system must be possible to be replaced with parts that are improved, more efficient, or that uses other algorithms, etc.
- M2. Modifiability: The interfaces between the parts of the system should remain the same when the system is changed.

Logical view:



The logical view is based on two architectural patterns: The model-view-controller pattern and the pipe-and-filter pattern. The model module holds representations of 2D scans in 2d model, a 3d model, a configuration representation of the system based on the parameters from the user, and a volume representation. In the filter module, the filter class contains the basic pipe-and filter functionality for transforming the input to some output. The dependencies of the various filters are modelled using the “uses” stereotype.

Development view:



Interfaces:
The interface between 2D models and the filters is raw graphic format 24bit (colour). The interface between the filters is also 24bit coloured raw format between Noise, Edge and B&W. The output of B&W is raw 2 bits format (black and white), which is the same format the Merger pattern outputs. The output interface of the Generator filter is the Universal 3D format (U3D). The interface between the Filters and GUI layer is the Universal 3D format.

The development view is shown as a layered architectural pattern separating the data, filters and the GUI. Note that some parts are left out in this view, like volume and parameters.