NTNU
Norwegian University of Science and Technology

Faculty of Informatics, Mathematics and Electronics

Department of Computer and Information Sciences

ENGLISH

Sensurfrist: 14. June

Exam in the subject
TDT4240 Software Architecture

Tuesday 24. May 2005
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:

Associate professor Alf Inge Wang, phone 73594485, mobile phone: 92289577

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 4 problems giving a total of 60 points.

Good Luck!

Problem 1 (10 points): Various questions

Answer these questions short:

1.1 Describe in your own words what a software architecture is?

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 difference between architectural patterns, design patterns, and idioms?

The abstraction level is different. Idioms are typically built into a programming language, design patterns are known solutions to recurrent problems refining a software component, and architectural pattern are typically known structures with known quality attributes.

1.3 What are the advantages of using the Abstract Factory design pattern in a software architecture?

The client does not know what variant it uses and this pattern is intended to support a family of related classes that can have different implementation details.

1.4 How can different stakeholders of a software system influence the software architecture? Give examples for Developer's organizations, Marketing, End-user, Maintainer, and Customer.

Different stakeholders have various interests of the architecture and need different views to see the parts they see as important.

- *Developer's organizations: Low cost, keeping people employed.*
- *Marketing: Neat features, short time to marked, low cost.*
- *End-user: Behaviour, performance, security, reliability, usability.*
- *Maintainer: Modifiability!*
- *Customer: Low cost, timely delivery, not chanced often.*

1.5 What is the relationship between functionality and quality attributes?

None. Functionality and quality attributes are orthogonal.

1.6 What is a quality attribute scenario and what is the purpose of it?

A quality attribute scenario is a tool to work out and document the quality wanted in the system.

1.7 What is an enterprise architecture and how does it relate to the business environment?

An enterprise architecture is an architecture at a higher level than software architecture that take into account IT-strategy, business goals and strategy, and enterprise portfolio. An enterprise architecture typically consists of strategy, projects, and IT system management. An enterprise architecture relates to the business environment through the overall business goals and strategies for the enterprise/company.

1.8 Describe the 3 main areas availability tactics must cover?

Fault detection, recovery-preparation and repair, recovery-reintroduction, and prevention.

1.9 What is Attribute-Driven Design (ADD)?

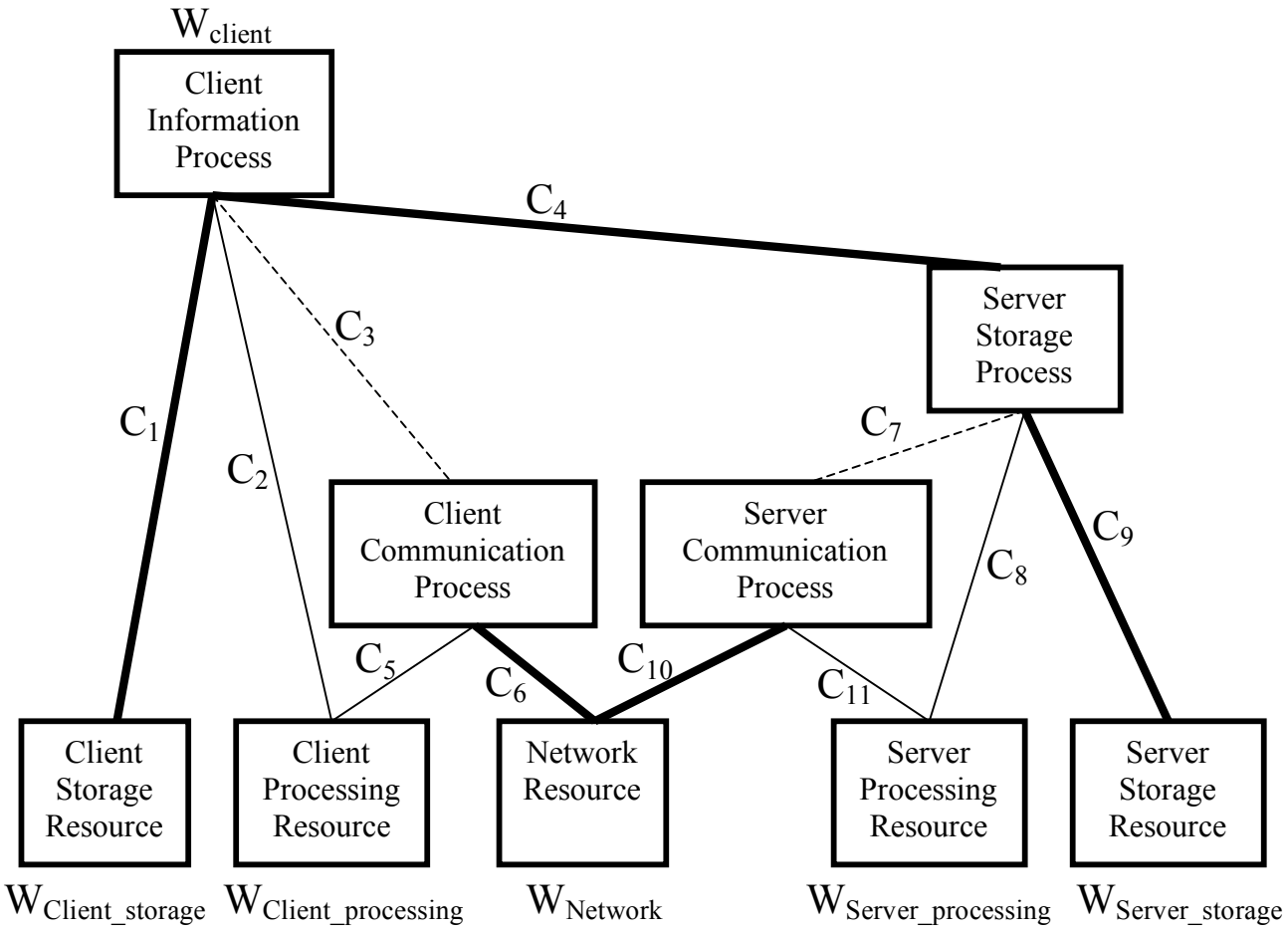
ADD is an approach to defining a software architecture that bases the decomposition process on the quality attributes the software has to fulfil.

1.10 Describe short the four steps of the reconstruction software architecture process.

- *Information extraction: Extract information from various sources (source files, files structures, documentation, developers, architects etc).*
- *Database construction: Convert extracted information to database tables (e.g. via Rigi Standard Format), typically by using Perl-scripts or similar.*
- *View fusion: Combine information in the database to produce a coherent view of the architecture (create new improved database tables (remove error sources and remove unnecessary information).*
- *Reconstruction: Build abstractions and representations to generate architecture views and documentations from the database.*

Problem 2 (10 points): Software Architecture and Performance

The figure below shows a Structure and Performance (SP) diagram.



Here are the operations (small circles) for the components in the diagram:

- Client Information Process: Read article, Write article.
- Server Storage Process: Select tuple, Update tuple.

- Client Communication Process: Send message, Retrieve message.
- Server Communication Process: Send message, Retrieve message.
- Client Storage Resource: Store block, read block.
- Client Processing Resource: Perform instruction.
- Network Resource: Send package, retrieve package.
- Server Processing Resource: Perform instruction.
- Server Storage Resource: Store block, read block.

Answer the following questions:

2.1 What does the diagram above describe and what is it used for?

The diagram describes a hierarchic relationship between resources with hardware resources at the bottom and software resources further up the resource hierarchy. The diagram shows how various resources are consumed in various parts of the system. Such diagrams are used mostly to analyse performance of a system e.g. detect bottlenecks, examine scalability. The diagram can also be used to visualise the system in a hierarchical manner.

2.2 What does the three different relationships mean (strong/bold line, normal line and dotted line)?

Strong lines denote memory relationships (links), normal lines denote processing relationships (links), and dotted lines denote communication relationships (links).

2.3 What is the purpose of the operations (small circles) in the diagram?

The operations (circles) describe typical unit of work performed in a component. E.g. for a communication process can typical operations be send and receive message. The operations make it possible to specify the workload of each component at the components specific level in the resource hierarchy.

2.4 What is the purpose of the Cs (C_1 to C_{11}) in the diagram?

The Cs are used to describe a complexity specification that describes the relationship between operations in two components. The Cs are matrixes where the content of the matrix can be functions, boolean, constants etc.

2.5 Find the work performed in $W_{Network}$ as a function of W_{client}?

$$W_{Network} = W_{client} \times C_4 \times C_7 \times C_{10} + W_{client} \times C_3 \times C_6 = W_{client} (C_4C_7C_{10} + C_3C_6)$$

Problem 3 (10 points): The ATAM

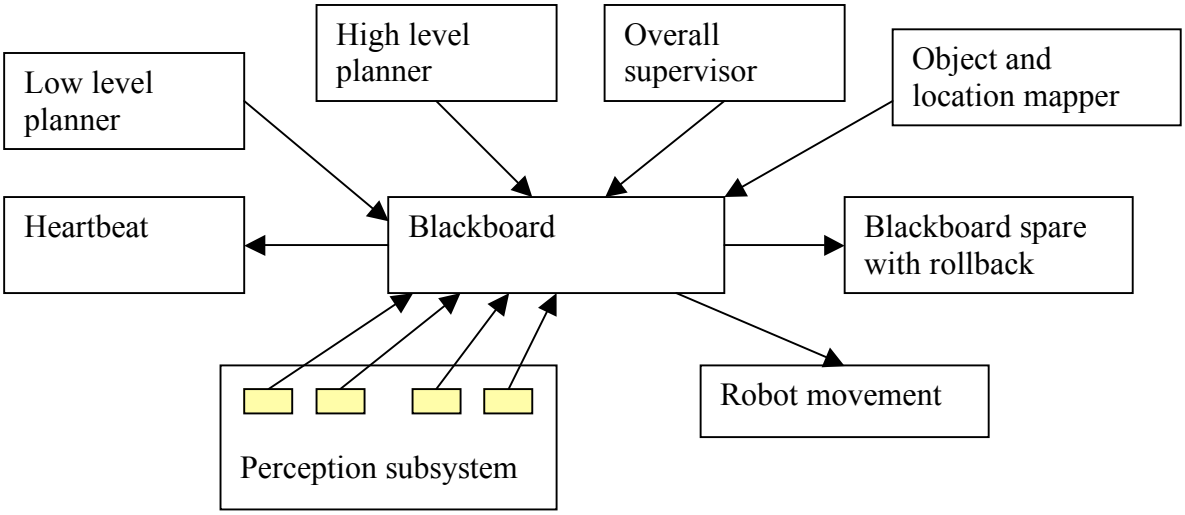
Read through the description of the robot controller architecture below and perform a *quick* ATAM evaluation of the architecture. Produce the following outputs:

- A utility tree
- Analysis of architectural tactics used
- List of sensitivity and tradeoff points
- Set of risks and non-risks
- Set of risk themes

Please motivate for your choices and state your assumptions:

Robot controller architecture

The focus of the architecture of this robot controller is on availability. A simplified illustration of the logical view is shown in the figure below:



Here is a short description of the different modules:

- *Overall supervisor*: Is the module that has the overview of the state and data of the robot and makes the decisions on what to do.
- *High level planner*: Plans high-level tasks like find ball, pick up ball, bring to light etc.
- *Low level planner*: Decomposes high-level tasks into movement commands.
- Object and location mapper: Identifies object and the surroundings of the robot.
- *Heartbeat*: Received control information from blackboard.
- *Perception subsystem*: Manages sensor information.
- *Robot movement*: Manages the motors of the robot.
- *Blackboard*: Manages the robot data.
- *Blackboard spare*: Manages the robot data.

The robot should be able to move around, pick up objects and bring the objects to the light. The downtime of the robot should not exceed 1 minute per hour, and if the robot controller goes down, it should be available within 5 seconds.

Utility tree:

- *Availability:*
  - *Scenario 1. The robot controller does not respond in normal operation, and notifies the user. The robot should be available 59 minutes of 60 minutes (98,3%). (M,H).*
  - *The robot controller does not respond in normal operation, and restarts/reinitiates the controller within 5 seconds (H,M).*

Identified architectural tactics:

- *Heartbeat – emits a heartbeat message periodically to tell that the component is alive.*
- *Spare –backup the data and the state of the controller in case of failure.*
- *Rollback – rollback data to a consistent state.*

Analysis of scenarios:

- *Scenario 1. Available 59 or 60 minutes:*
  - *Heartbeat: S1, R1*
  - *Spare: T1, R2*
  - *Rollback: T2, N1*
- *S1. Positive for availability by checking if the system is alive.*
- *T1. Positive for availability by backing up data. Negative for performance on extra transfer of data.*
- *T2. Positive for availability to go back to consistent state. Negative for performance on replication of state.*
- *N1. Rollback is safe as long as the state has been stored.*
- *R1. Need extra independent hardware or separate process that not will fail with the controller.*
- *R2. Need extra independent hardware or separate process that not will fail with the controller.*

Risk themes: To provide availability it is very important that the backup and heartbeat mechanisms do not crash or go down with the controller.

Problem 4 (30 points): Create an architecture

Read the description below and do the following:
4.1 Identify the most important quality attribute(s) for the system described below.
4.2 Identify architectural driver for the system described below.
4.3 Choose and describe suitable architectural tactics for the problem described below, and describe how the tactics affect the quality attributes.
4.4 Create architecture views of the system described below. The architecture must be described in two views according to the 4+1 view model: *Scenario view and Logical view*.

Motivate for your choice of quality attributes, architectural drivers and the architectural tactics used in your architecture.

Software for Digital Camera

The software described here is software that is used in the controller of various digital cameras. The software should be able to provide different levels of functionality depending on the price segment of the camera and the software should be able to be used for various kinds of hardware configurations (buttons, screen, data storage and optical components).
The digital camera consist of these hardware components:

- Controller (CPU, memory): managing the other components, provide interface with the user etc.
- Controller buttons (vary from camera to camera). Typically buttons for on/off, flash, take picture, menu, navigation, zoom etc.
- Digital screen (can vary from camera to camera in size, colour depth etc).
- Permanent data storage (typically flash-memory, memory stick, SD-cards etc).
- Optical component with an interface to control zoom, focus, etc....

Here is a list typical functionality provided by the camera:

- Turn on/off camera.
- User controlled optical functionality (zoom out, zoom in, flash etc).

- Camera controlled optical functionality (auto focus, lens opening etc).
- Power save functionality (shut down camera if not used etc).
- Storing, retrieve and delete pictures.
- Processing images (rotate, enhance, etc).
- Display pictures, information to the user on the camera's screen.
- Camera set up (storage options, GUI-options, language options etc.)

4.1 Most important quality attributes: Modifiability, availability and performance are the main quality focus areas of the system. For modifiability it is important that you can use various kinds of hardware components (buttons, screens, data storage and optical components). For availability it is important that the software does not crash and the user of a camera expect no failure. For performance, it is important that the camera is responsive so it does not take too long before the user can take pictures or use the functionality of the camera.

4.2 Architectural drivers for this system are: Variation in hardware components and configurations, and availability of the software.

4.3 Architecture tactics:

4.3.1 Modifiability: Maintain semantic coherence, especially on the variation points for hardware components and interfaces. Hide information. Divide functionality of the camera into building blocks that can be added or removed on configuration.

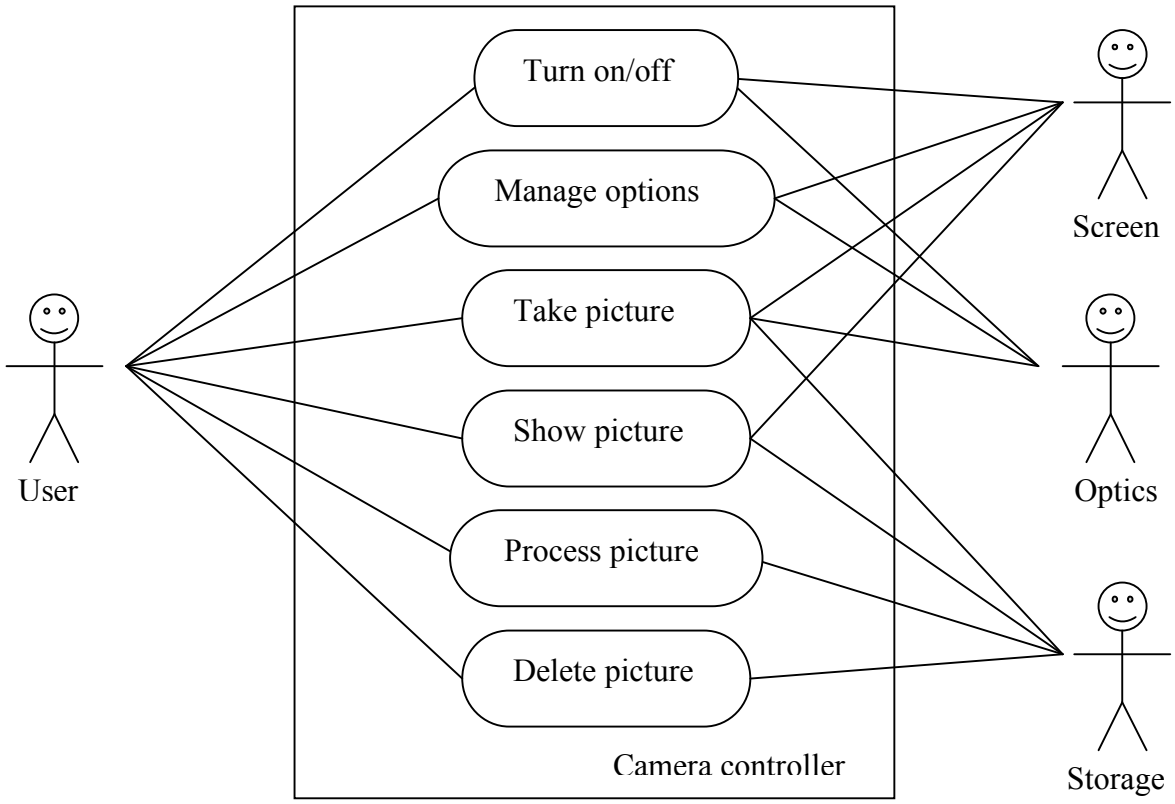
4.3.2 Availability: Exceptions, heartbeat, checkpoint/rollback.

4.3.3 Performance: Schedule performance critical functionality.

4.4 Software architecture views:

Note that there is not one correct diagram to this problem, but rather a lot of possible solutions. For the scenario view, it is important that the most important functionality of the camera and the most important actors are shown. For the logical view, it is important that this view visualise the modifiability/extendibility of the system (e.g. that the architecture should be possible to be used for various configurations).

Scenario view: The scenario view shows the main actors (User, screen, optics and storage) and the main functionality of the camera.



Logical view: The motivation for the logical view is to have good support for productline and modifiability. Dig.scr = digital screen, Opt.dev = optical device, Sto.dev = storage device. The extra functionality is divided into simple, medium and advanced to distinguish between different price segments of cameras with different level of functionality.