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: 8. June

Exam and solution in the subject TDT4240 Software Architecture

Friday 18. May 2007
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 5 problems giving a total of 70 points. The remaining 30 points are credits awarded from the robot project.

Good Luck!

Odd Petter Slyngstad

Controlled 8th of May 2007

Problem 1: Various questions (15 points)

Answer these questions short:

Second.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.

Second.2 How are quality attributes related to software architecture?

You can achieve certain quality attributes in a system by selecting the suitable software architecture. E.g. some architectures will higher availability etc. However, quality attributes are also dependent on other things like code quality, quality of work, choice of algorithm etc.

Second.3 How can a software architecture benefit from using design patterns?

Design patterns contribute as tactics to achieve qualities in certain areas. The design pattern ensure that the architecture achieve the quality on the design level.

Second.4 What is architectural erosion?

The architectural documentation is frequently updated to reflect the implementation until the architecture has no clear structure or direction.

Second.5 How can the architecture be influenced by how the company is organized (organization structures, departments, size, etc.)?

The organization of the company reflects how the architecture of the system is structured to reflect what parts that can be implemented in parallel; the knowledge of departments and how many that should work on different part of the system.

Second.6 Describe the differences and the relationships between reference model, architectural pattern, reference architecture and software architecture.

Reference model: A division of functionality together with dataflow between the pieces. Architectural pattern: Description of element and relation types together with a set of constraints on how they may be used. Reference architecture: Reference model mapped onto architectural pattern. Architecture: Specified reference architecture to solve problem for specific customer.

Second.7 Explain the difference between business quality, architecture quality and system quality according to the textbook (Bass, Clements and Kazman).

Business quality is characteristics related to business goals of the system such as time to market, cost, system lifetime, targeted marked, rollout schedule, integration with legacy systems. Architecture quality is characteristics of the architecture itself such as conceptual integrity, correctness, completeness and buildability. System quality is characteristics about the software system.

Second.8 What is the relationship between tactics and architectural patterns?

Architectural patterns consist of a collection of tactics to achieve certain qualities.

Second.9 What are the typical relations that you can extract when reconstructing software architecture (e.g. in the programming language C)?

Typical relations that can be extracted are file include file, file contains function, file defines variables, directory contains directory, directory contains files, function calls function, function access variables (both read and write).

Second.10 What is the problem with too narrow or too broad scope for a software product line?

Too narrow: Insufficient number of products will be derived to justify investment (too specific). Too broad: Effort to develop individual products will be too grate (too general).

Second.11 Name the main forces that affect a game architecture?

New consoles/hardware, new genre of games, online gaming, wireless gaming, increased collaboration between engineers and artists, and multi-platform.

Second.12 What is a service-oriented architecture?

A service-oriented architecture is an infrastructure that structures a set of services in such a way that it can meet the needs, be combined and be used of the business. The architecture must be flexible, so it is easy to combine the various services in a way the give a business value.

Second.13 What is a quality attribute scenario and what is the purpose of it?

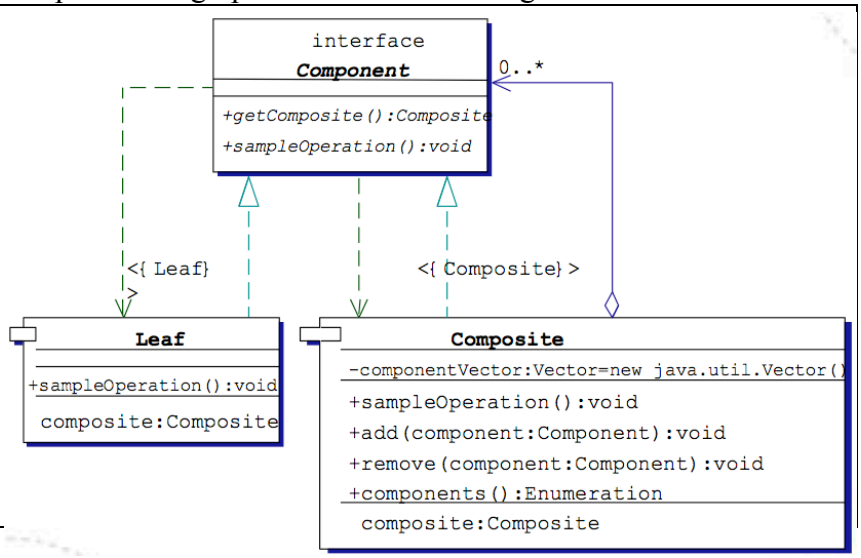
A quality attribute scenario describes a specific quality requirement that a final system should have in such a way that it can be tested/evaluated. A quality attribute scenario is described by source of stimulus, stimulus, an artifact, the environment, response and a response measure.

Second.14 What are the outputs of the ATAM?

Main outputs: Mapping of architectural tactics to quality requirements/scenarios, a set of identified sensitivity and tradeoff points, a set of risk and nonrisks and a set of risk themes. In addition ATAM will enforce a concise presentation of the architecture, an articulation of the business goals/architectural drivers, and quality requirements in terms of a collection of scenarios.

Second.15 Describe the Composite design pattern and its advantages?

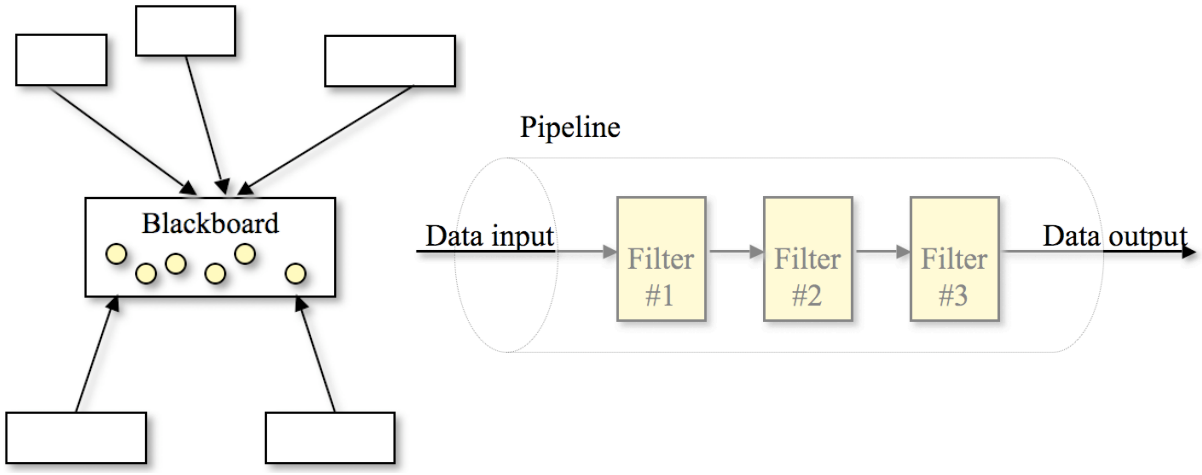
The composite design pattern is used to represent hierarchies of objects where the objects can be manipulated in the same manner whether the target objects are individuals or groups. This gives a transparent treatment of single and multiple objects.



Problem 2: Architectural patterns (10 points)

2.1 Compare two architectural patterns (5 points)

You are going to create the architecture for a robot controller where the main focus is on *modifiability*, so you easily can change the behaviour of the robot. ***Compare the two architectural patterns*** Blackboard and Pipe and filter (as shown below) and ***choose the pattern that best fits the quality requirement*** (motivate for your choice).

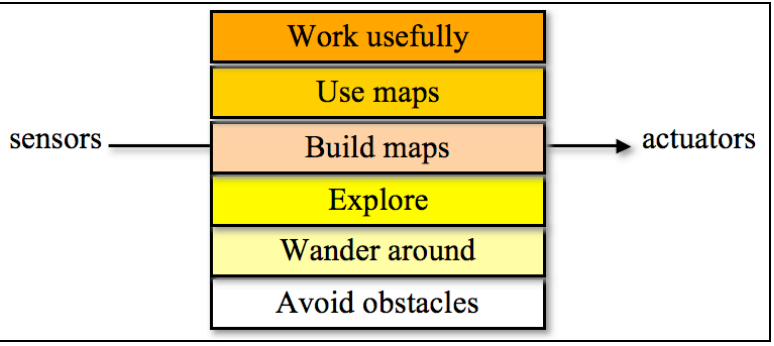


Pipe and filter is suitable to manipulate a datastream by converting the data input for each filter. This pattern gives a simple process flow, but the system will not be very flexible in terms of modifiability because of the tight coupling through the datastream. E.g. if Filter #1 is changed, the modification might cause a change in both Filter #2 and Filter #3 as they depend on Filter #1.
The Blackboard pattern relies on a central database where all components can publish and subscribe information objects. This pattern gives a very loose coupling between the components making it possible to add and change components easily. The blackboard pattern enables dynamic/run-time modification of the robot controller.
Conclusion: The blackboard architectural pattern should be used to achieve high modifiability.

2.2 Describe the Subsumption Reference Architecture (5 points)

Describe the subsumption reference architecture using text and a figure, and describe the characteristics for this reference architecture.

Layered pattern to adds more and more complexity at higher layers. Each layer implements one behaviour. The complexity is decomposed into layers. Fits well with incremental development approach. This architectural pattern is bound to behaviour.



Problem 3 (10 points): The CBAM

From the Table 1, 2 and 3 and the information below, find (use *straight lines between the data points in the graph*):

- Total benefit obtained from the 3 architectural strategies.
- Return-On-Investment for the 3 architectural strategies
- Rank the 3 architectural strategies according to best investment.

The data is results from applying the Cost Benefit Analysis Method (CBAM) on a web-system for selling tickets for cinemas, theatres, music concerts, and sports events.

Total Benefit Obtained can be computed using this formula: $B_i = \Sigma (b_{i,j} \times W_j)$ where:

- B is benefit for each architectural strategy *i*
- $b_{i,j}$ is the benefit by using strategy *i* to its effect on scenario *j*
- W_j is the weight of scenario *j*

Table 1: Results from prioritizing scenarios with worst, current, desired and best response levels.

Scenario	Vote	Worst	Current	Desired	Best
1. Performance (Time to pick up and deliver 4 balls to the light)	20	60min	20min	5min	3min
2. Availability (How many times the robot get stuck during the mission picking up balls)	20	50 % fail	20% fail	5% fail	0 % fail
3. Availability (How many times the robot controller software crashes)	35	30% crash	10% crash	0% crash	0% crash
4. Modifiability (Time to add readymade module)	15	8hours	60min	10min	5min
5. Testability (Time to test robot movements)	10	10hours	5hours	30min	10min

Table 2: Results from assigning utility to the various scenarios.

Scenario	Vote	Worst	Current	Desired	Best
1. Performance	20	10	50	90	100
2. Availability	20	10	55	85	100
3. Availability	35	15	75	100	100
4. Usability	15	0	50	70	100
5. Testability	10	35	60	80	100

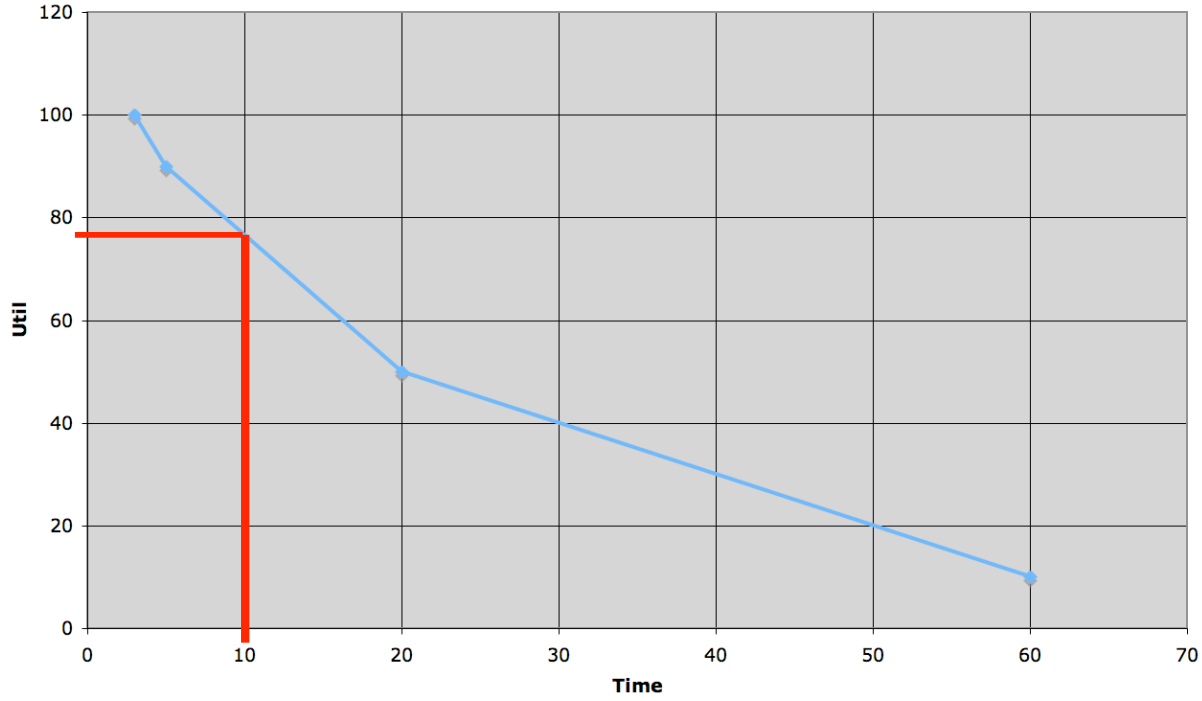
Table 3: Effect and cost of using architectural strategies.

Strategy	Scenario	Cost	Current response	Expected response
1. Use route-planning	1	2000	20min	10min
2. Improved wall detector	2 3	800	20% fail 10% crash	7% fail 10% crash
3. Improved exception handling	2 3	1100	20% fail 10% crash	20% fail 2% crash

Solution:

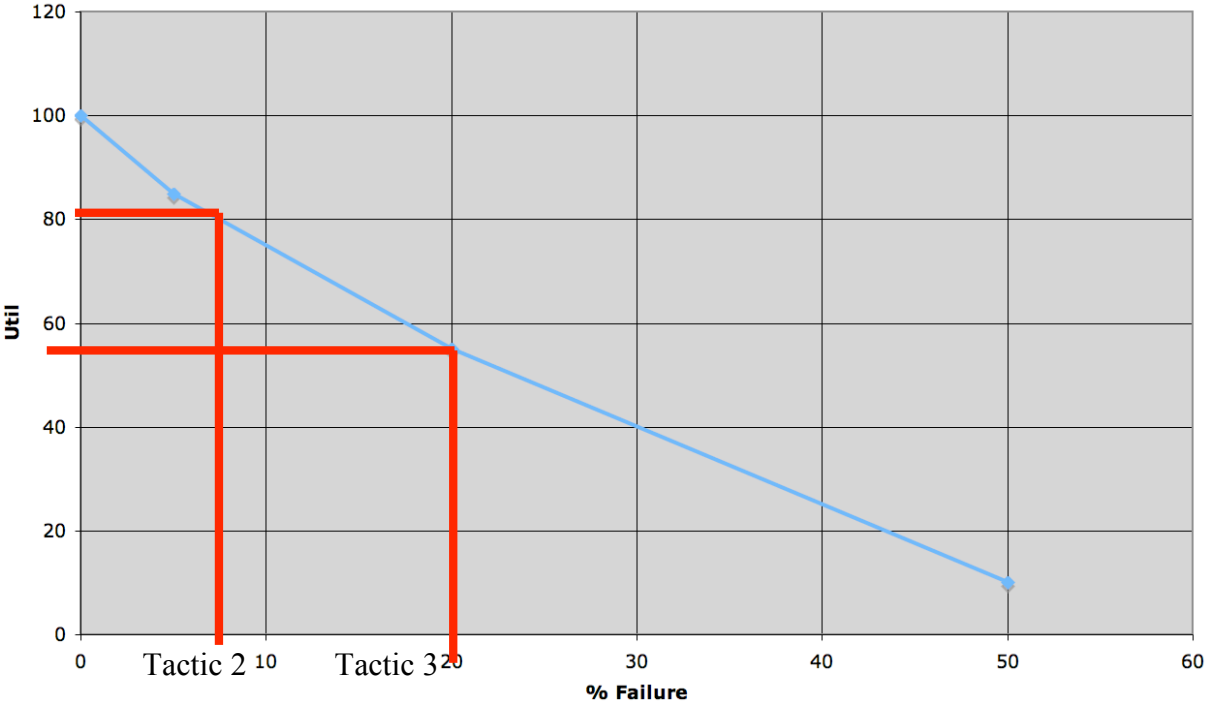
To compute the total benefit obtained, the return-on-investment, and find what architectural strategy is best investment, utility response curve for scenarios 1-3 must be made (scenarios 4-5 can be ignored as there is not defined any tactics that will affect them). From these curves the utility of expected response when applying the architectural strategy can be read from the graph. Note that the numbers can be slightly different depending on how the graphs are drawn. However, the conclusion should be the same.

Utility response curve scenario 1:



- Using tactic 1 on scenario 1: 10 minutes pickup time gives 77% utility.

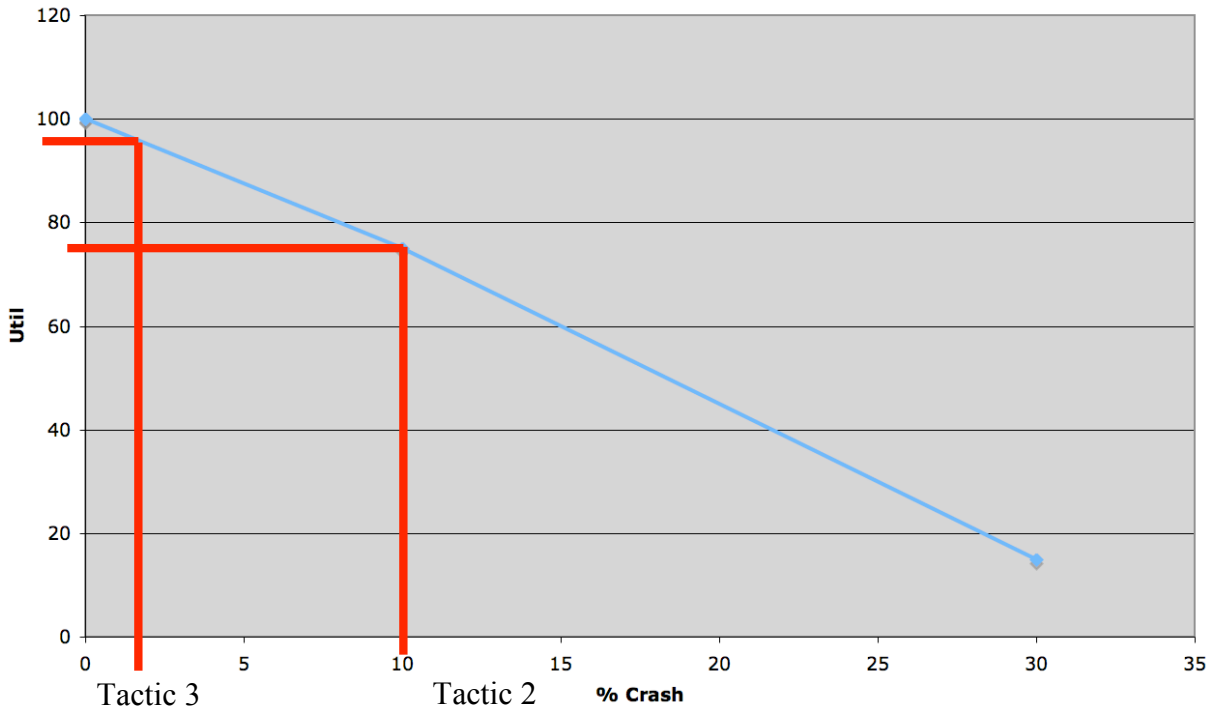
Utility response curve scenario 2:



- Using tactic 2 on scenario 2: 7% fail gives 81% utility

- Using tactic 3 on scenario 2: 20% fail gives 55% utility

Utility response curve scenario 3:



- Using tactic 2 on scenario 3: 10% crash gives 75% utility
- Using tactic 3 on scenario 3: 2% crash gives 95% utility

Total benefit on tactics 1, 2 and 3:

B_1 = (77-50) x 20 = 540
B_2 = (81-55) x 20 + (75-75) x 35 = 520
B_3 = (55-55) x 20 + (95-75) x 35 = 700

Return-on-investment on tactics 1, 2 and 3:
ROI_1 = 540/2000 = 0,27
ROI_2 = 520/800 = 0,65
ROI_3 = 700/1100 = 0,63

Ranking of tactics (from best to worst):
1 Tactic 2: Improve wall detector
2 Tactic 3: Improve exception handling
3 Tactic 1: Use route-planning

Problem 4: Reconstruction of a software architecture (5 points)

Read the description of the software system below and *suggest a practical approach for reconstructing the architecture of the system*. Describe the information you will use in the reconstruction and how you will use it.

The company HugeSoft™ implemented the banking system BankBankWhoIsThere (BBWIT) in 1986 and several banks have used this system since then. HugeSoft wants integrate this system with an insurance system and need to establish a software architecture to see if it is possible and how. No software architecture or design documentation exists for the system, and none of the architects or designers of the system are accessible. The only artefacts you have available from the system are the 8,000 files of source code written in the programming language C distributed in more than 200 directories. The system consists of more than 2,000,000 lines of source code, and most parts of the system are divided into modules that usually include hundreds of files.

Suggested solution:
The characteristic of this system is that it is a huge system, and thus a detailed analysis would be too time-consuming. The best approach is to start analysing the directory relationships (directory in directory, and files in directory) to see if this can reveal a kind of modular structure of the system. For further analysis, the source code can be used to analyse function and variable dependencies. Since there is so much source code, this work must be possible to be automated and the results must be stored in a database. Then it is important to try to abstract the information wherever possible to get an overview of the system. Dynamic analysis of the system is not recommended as the goal is to integrate with other systems and it would be very tough to perform a run-time analysis on such a big system.

Problem 5 Create an architecture (30 points)

Read the description below and do the following:

5.1 Identify the ***architectural drivers*** for the system described below (5 points)

5.2 Choose and describe suitable ***architectural tactics*** for the problem described below, and describe how the tactics affect the quality attributes (5 points)

5.3 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 Process view* (20 points)

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

Software for SoNewHenriksen$_{TM}$ Mobile Phones

The software described here is software for operating various models of SoNewHenriksen$_{TM}$ (SNH) mobile phones. The different models vary in the type of data communication supported, display, keypad and buttons, processor and the type of functionality offered. A SNH mobile phone consists of the following physical parts:

- A mobile processor with memory, Flash ROM and storage (varies from model to model).
- A display (varies from model to model)
- Buttons and keypads (varies from model to model)
- A communication unit responsible for all communication (varies from model to model)
- A loudspeaker (same for all models)
- A microphone (same for all models)
- A battery (same for all models)
- A connection interface for charging and connection to other units like PCs (same for all models)

The software for operating the SNH mobile phones provides the following basic services that are the same for all phone models:

- Call-functionality (e.g., make phone calls, list of callers etc.).
- Address book.
- Messaging functionality (e.g., SMS, MMS, Email, etc.).
- Time, Calendar and Alarm.
- Setting/Preferences.
- File management.

In addition SNH mobile phones can offer additional functionality (varies from model to model) through addition applications that can be pre-installed on the phone:

- Internet functionality (e.g., web-browser).
- Multimedia functionality (e.g. audio player, video player, radio, etc.)
- Camera functionality.
- Games.
- Etc.

Proposed solution

5.1 Architectural drivers

The main architectural driver for this system is modifiability (both hardware and software) as this is a product line system with a lot of variation.
Another important architectural driver is availability (especially the main functions such as call-management) as a phone should be possible to be used in emergency situations.
Performance is also an important architecture driver as mobile devices have limited resources in terms of CPU, memory and storage.
For the user of the system, usability of the mobile device is important because of the limitations of small screen and keyboard.
Related to business issues, time to market and thus short development time is critical to have a new model of mobile phones out before other companies.

5.2 Architectural tactics

The main focus here should be on modifiability as this is the main architectural driver:
- Use Model-view-controller architectural pattern to allow separation of user display, user input and logic.
- Use generic classes with extendable sub-classes to support variations in keyboard, display and communication.
- Use virtual machine to support various processors.

Related to availability, the critical call-functionality and communication modules should get extra priority in terms of reliability in case other functionality crashes. The software should have extensive failure handling that can reboot the software in case it hangs (fast reboot).

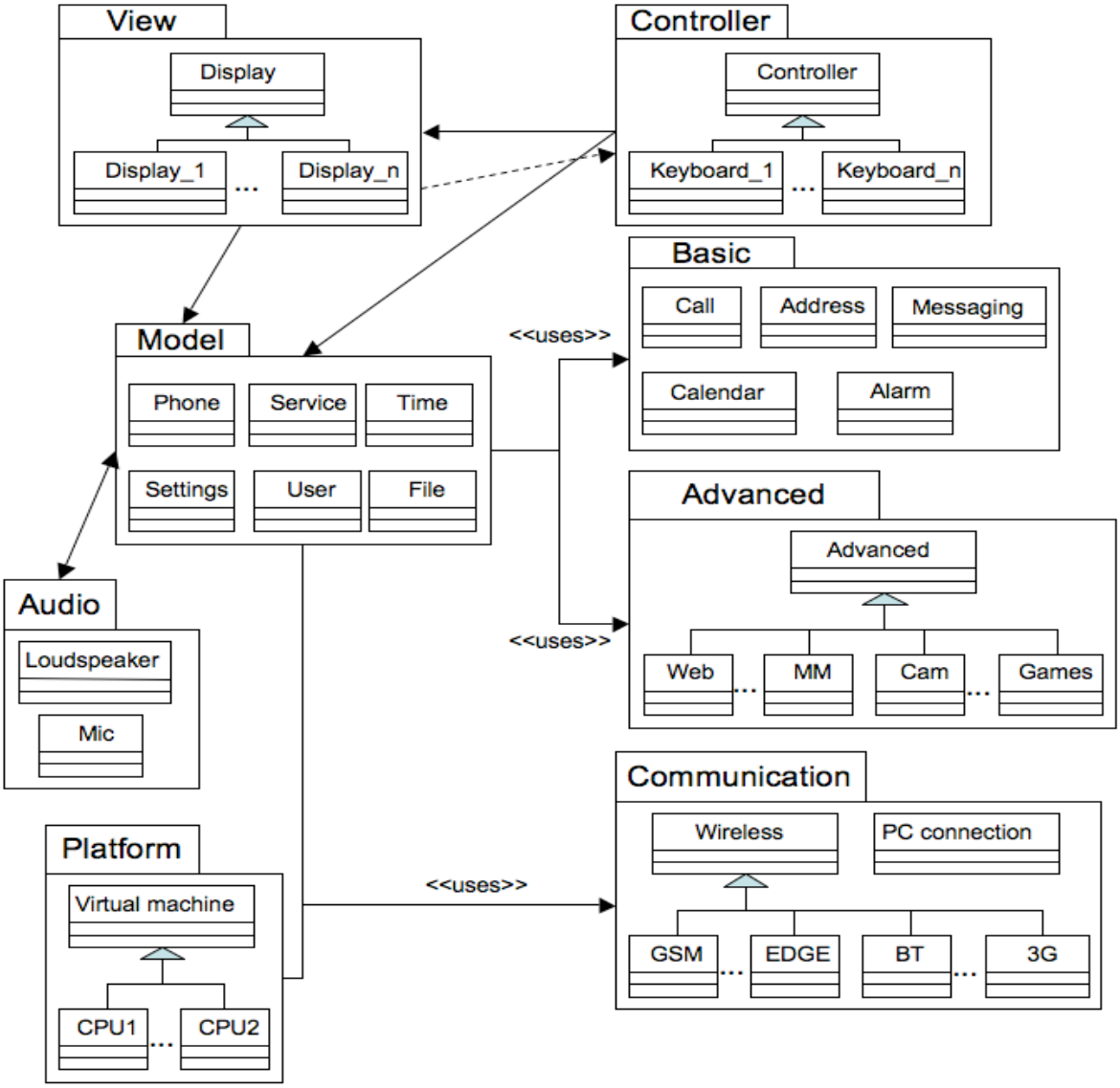
Related to performance, the whole system should be designed to be memory and CPU efficient to avoid slow performance.

Related to usability, the model-view-controller architecture pattern will provide a flexible solution to provide flexible user-interface that can be changed. Also the user interface should provide context-sensitive by maintaining a model of the user to provide help-functionality related to what the user is doing.

5.3 Architecture views

The logical view reflects the focus on modifiability by using the Model-view-controller pattern as well as using variation points for display, keyboard, advanced functionality, CPU configurations and communication networks. The variation for memory and storage is not visualised in the logical view as it is expected that the operating system takes care of this part. The model maintains logical representations of the phone (phone state etc.), settings (preferences), user (for usability), file management and services running. The services available is based on the two modules Basic and Advanced. The services in these modules can be invoked and the management and the state of the service is maintain within the model module.

Logical view:



Process view:

The system will have three main processes (threads) that will run in parallel. The first process, *model*, is responsible for maintaining a runtime statue of the system (phone status, time, service status, user status, settings, file etc) or coordinating other processes on the phone. The second process, *communication*, is responsible for handling the communication and checking for incoming calls, messages etc. The third process, *service*, is the current service invoked by the user and can be one of the basic services (like call, address, calendar, messaging or alarm), or advanced services (like Web, Multimedia, Camera or Games).