NTNU
Det skapende universitet

Department of Computer and Information Science

SOLUTION for TDT4240 Software Architecture Exam

Academic contact during examination: Adjunct associate professor Gunnar Brataas
Phone: +47 47 23 69 38

Examination date:	Wednesday 28 May 2014
Examination time (from-to):	09:00 – 13:00

Permitted examination support material:

- 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 an English thesaurus (English-English).

Other information:

- Simple calculator or a calculator approved by NTNU allowed.
- The exam has 3 exercises giving a total of 70 points. For each exercise, each question has the same weight unless otherwise stated. The remaining 30 points are credits awarded from the software architecture project.
- If there are mismatches between the English version and the Bokmål or Nynorsk version, then the English version is superior.

Language: English
Number of pages: 7

Exercise 1: AMS Architecture (30 points)

Figure 1 shows an Advanced Metering System (AMS). A Consumer has electric Devices like water heaters, electric heaters or washing machines. An Advance Meter measures the energy usage of all the Devices of the Consumer, and sends this to the electric energy Distribution Organisation. The Distribution Organisation uses this information for billing. The Distribution Organisation also sends Price Signals to the Control System, telling when electricity has a high or a low price. Depending on the configuration of the Consumer, the Control System can use these Price Signals to turn on or off Devices. For example, a water heater may be turned off when energy is expensive at daytime and turned on when energy is chap at night. This will reduce the electricity bill for the Consumer, and will also give a better balance between energy production and energy demand in the whole electricity energy grid, which will make it easier to utilise green energy sources like wind mills, solar power etc.

The AMS will consist of both the Advance Meter as well as the Control System and will interact via the Internet with the Distribution Organisation. In addition, the AMS will use an internal network to communicate with the Devices. The Advanced Meter and the Devices are connected via physical electrical cables. The software in the Distribution Organisation or the software in the Devices may change in the future.

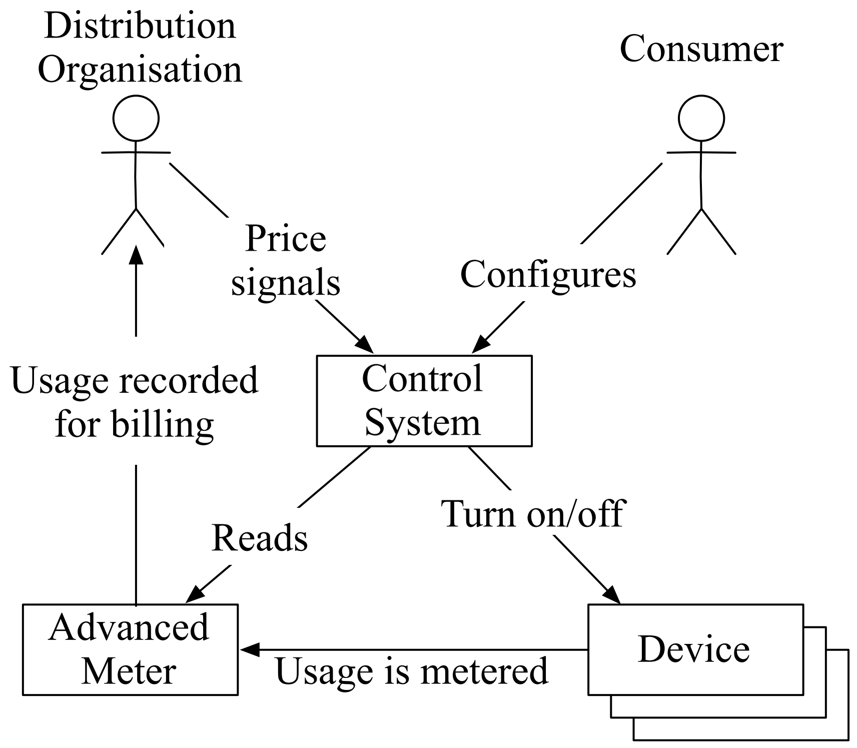

Figure 1 Major Components of an AMS.

a) (4 points) Why is modifiability important for the AMS?

Answer:

AMS must adapt itself to the changes in the software in the Distribution Organisation and software in the Devices.

b) (8 points) Security has the characteristics confidentially (hint: data access) and integrity (hint: data manipulation). Describe an AMS scenario for each of them. Remember that there are six portions of a scenario.

Answer:

Half a point for each of the six portions for the two characteristics, plus one point for very good descriptions on either confidentiality or integrity:

Confidentiality: A hacker (source) can at run time (environment) using the meter information which he reads (stimulus) from the Internet communication (artefact) between the AM and the Distributor Organisation to find out which customers are currently on vacation. He can then break into those houses. As a *response* to this threat, the communication can be encrypted. A *response measure* is how likely it is to read meter data after encryption. For example, we can have a response measure that only accepts a 100% success for test cases to break the encryption data. Information about the encryption algorithm can also given hints on how safe it is.

Integrity: A neighbour (source) can at run time (environment) increase (stimulus) your metering information (artefact) and reduce his metering information by the same amount, so that his electricity bill is reduced and your electricity is increased. As a *response* to this threat the communication can be encrypted. A *response measure* is how likely such tampering becomes after encryption. This is not easy to measure. For example, we can have a response measure that only accepts a 100% success for test cases to break the encryption data.

Note: Other plausible ways of answering this question should also get full score.

c) (6 points) List six different ICT stakeholders in the AMS.

Answer:

One point for each correct stakeholder (see page 54-55 in Bass for inspiration), with a maximum of six points. Below we list eight potential AMS roles:

- Distributor Organisation
- Consumer
- Architect
- Analyst
- Developer
- Tester
- Maintainer
- Product manager

Note: Some flexibility with other names. Therefore, one point can also be given for other stakeholders, given that it seems plausible and are not too overlapping with other stakeholders.

d) (8 points) We now focus on the eleven architectural patterns described in the Bass book. List the two most suited architectural patterns for AMS and describe how they fit.

Answer:

Each correct pattern name will give two points each and each convincing argumentation will give two points.

The following patterns are natural to use:

- MVC: the consumer needs a UI, which is then a View. The Model is likely to be stored in the AM and the AMS Control System (except for the UI) will be the MVC controller.
- SOA: the Distribution Organisation offers services to the AMS.

Note: Other plausible patterns can also be accepted, for example:

- Layered: we will at least have an OS and a UI. In addition, more layers for network communication etc. are also likely.
- Client-server: the distribution organisation is the server and the AMS is a client. (SOA is a specialisation of client-server.)

Publish-subscribe, multi-tier, pipe-and-filer and broker could also be argued for. However, map-reduce, peer-to-peer and should be hard to justify.

Without an argumentation, plausible patterns will of course only get one point each. It is not possible to get more than four points for pattern names.

e) (4 points) Describe two possible deployment (physical) views for the Control System in Figure 1 (Hint: the Control System requires a user interface (UI) to communicate with the Consumer).

Answer:

Two points for each correct deployment.

Alternative one (conceptually simplest): All of the software modules in the Control System run on one physical CPU (which are likely to be placed in your fuse box).

Alternative two (most likely): The configuration UI part of the Control System will be placed separately from the rest of the Control System. This configuration UI can be placed in a separate device in your living room, or it may be an application on your smart phone.

Exercise 2: Theory (25 points)

a) (8 points) List the four basic architectural drivers, each with an example from AMS.

Answer:

One point for each driver and one point for each good example

- Functional requirement: e.g.,
    - the Control System use Price Signal to turn on or off Devices and
    - the Control System requires configuration from the Consumer.
- Quality requirement: e.g., the meter information should be reported once every minute.
- Business requirement: Energy/cost saving capability and profitability
- Technical requirement: e.g., a particular communication protocol shall be used to communicate with the Distribution Organisation.

b) (7 points) What are the seven basic outputs of the ATAM?

Answer:

See Bass page 402-3:

1. A concise presentation of the architecture
2. Articulation of the business goals
3. Prioritises quality attribute requirements expressed as quality attribute scenarios
4. A set of risks and non-risks
5. A set of risk themes
6. Mapping of architectural decisions to quality requirements
7. A set of identified sensitivity and trade-off points

c) (4 points) What is meant by the "connector" in the "component and connector view"? Give at least one good example.

Answer:

Bass page 335-340 for connector definition; page 210-231 for examples of connectors in the patterns.

Two points for definition and two points for a good example.

Connectors are pathways of interaction between components. Connectors have roles or interfaces that indicate how components may use a connector in interactions.

Examples of connectors are pipes or asynchronous message queues.

d) (6 points) How can we analyse a documented software architecture to see if it fulfils the following quality attributes: performance, availability, and security? What is analysis paralysis and how can it be avoided?

Answer:

See Bass Table 14.2 on page 259, plus page 264. 1,5 point for each complete answer:

- Performance: using the documented software architecture plus some more information we can produce performance models that can give good insight into performance challenges for the system. In some cases, we can also get good insight using very simple back of the envelope models. The models will typically be queueing network models.
- Availability: like for performance we can build availability models using the software architecture models supplemented with some more information concerning availability. The models used will typically be Markov models.
- Security: checklists can be applied with a low cost.
- Analysis paralysis: spending endless time trying to analyse an architecture. Can be avoided by setting a deadline on discussions and by looking at the trade-off between cost of discussing and cost of the problem that may be solved.

Exercise 3: Software Product Lines (15 points)

a) (2 points) Define a software product line (SPL).

Answer:

Bass page 479: 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.

Note: The closer to this definition, the more points. The core is that several products are developed from a common core, using variation points.

b) (3 points) What are the three primary architectural variation mechanisms?

Answer:

Bass page 490-1:

- Inclusion or omission of elements
- Inclusion of different number of architectural elements
- Different versions of the an elements with compatible interfaces

c) (4 points) Describe the trade-off between a narrow and broad scope of an SPL?

Answer:

See Bass page 486-8:

A narrow focus gives few potential products and few and simple variation points, whereas a broad focus gives too much effort in designing products based on the core and many and complex variation points. One should find communalities that can be economically exploited to reduce the cost of making individual products based on the core.

d) (6 points) Describe the three adaption decisions when introducing an SPL approach in an organisation.

Answer:

See Bass page 494-6:

Only to name them gives one point each. In addition, each good description gives one point.

- Top-down versus bottom-up adoption: Top-down is when SPL adoption is initiated from the top in an organisation and it is expected that the programmers, architects etc. change the way they work. Bottom-up is when programmers, architects etc. that design slightly different systems, convince management to introduce SPL.
- Proactive versus reactive adoption: The proactive approach involves planning and can therefore be more far reaching, but will also be more complex to implement. The reactive approach is in line of the agile approach and requires less planning, and is therefore easier to implement.
- Incremental versus big-bang adoption: Using the big-bank approach the core components are determined initially, which means the organisation must focus all the effort on this for a period. Using the incremental approach, the size of the core components slowly increase, so that the first products are developed using a smaller set of core components than later products.