NTNU – Trondheim
Norwegian University of
Science and Technology

Department of Computer and Information Science

SOLUTION for Examination paper for TDT4240 Software Architecture

The English version is the reference version of the examination paper!

Examination date: Tuesday June 11th 2016

Solution Problem 1 (10 points)

1.1 The software architecture of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both.

1.2 Reduce size of a module, Increase cohesion, Reduce Coupling, Defer binding.

1.3 Reasons why software architecture is important: Tool to realize a system's quality attributes, reason and manage changes in a system, prediction of a system's qualities, enhances communication among stakeholders, careful planning of most fundamental hardest-to-change design decisions, defines constraints on implementation, dictates structure of an organization and vice versa, basis for evolutionary prototyping, enable reasoning about cost and schedule, enable evaluation of system before implemented, provide a reusable model that can form a product line, reducing design and system complexity, foundation for training...

1.4 The purpose of architectural views is that it makes it possible to address concerns of different stakeholders with varying interests. It is also impossible to cover all the complexity of a software architecture in one single view.

1.5 A sensitivity-point describes how a design decision will affect only one quality attribute (usually positively), while a tradeoff point describes how several quality attributes are affected both positively and negatively.

1.6 CBAM includes an economical analysis and is used to find the architectural decisions within the resources available.

1.7 CBAM process:

Collate scenarios
Refine scenarios (worst, best, current, desired)
Prioritize scenarios
Assign utility to current and desired levels
Develop architectural strategies and determine quality attribute response levels
Determine expected utility value of architecture strategy using interpolation
Calculate total benefit for architectural strategy
Chose architectural strategies based on Value for Cost
Confirm results with intuition.

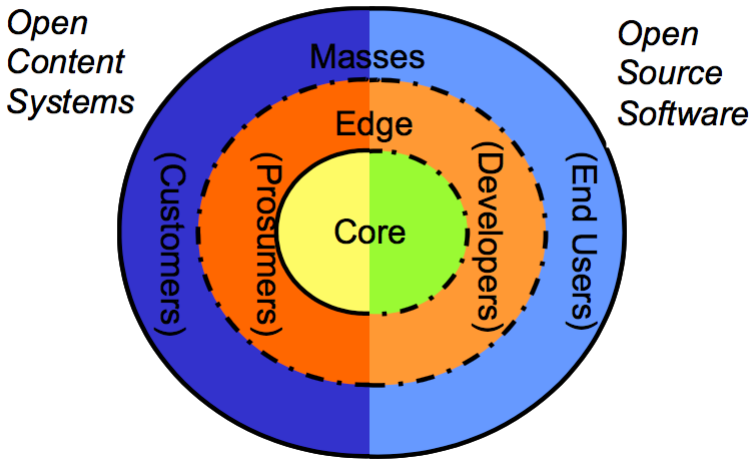
1.8 A game loop where the drawing of frame is not done until the AI and collisions are calculated is likely to have a unstable frame-rate depending on the number of enemies and the situation in the game, compared to a game-loop with full decoupling of the AI and collision calculations.
1.9 Context, Problem, Solution
1.10 Creational: Factory, Singleton
1.11 Structural: Composite, Proxy, Façade
1.12 Behavioral: Observer, State, Strategy
1.13 Statistical or Markov models (probability models)
1.14 Cost in terms of number/size/complexity of affected artifacts, Cost in terms of effort, Cost in terms of time, Cost in terms of money, Cost in terms of the extent modifications affect other functionality or quality attributes, Cost in terms of introduction of new defects.
1.15 Reconstruction process: Information extraction, Database construction, View fusion, Reconstruction of the architecture
1.16 Enable maintenance/integration/expansion/reuse of legacy system, enable comparison of different systems, check conformance against as-design architecture.
1.17 A designed architecture is a planned architecture (as-design architecture), while a reconstructed architecture is the architecture as the system was implemented (as-is architecture)
1.18 When you want to develop software for many similar products with some variations but that share some common reusable features.
1.19 Give a structured overview of the quality requirements organized according to quality attributes.
1.20 If need to be able to operate a single object and multiple objects with the same operations.

Solution Problem 2 (5 points)

1. a) Layered: The example consists of components of different abstraction levels, from the hardware abstraction level, the core basic functionality, to the higher level component with an application API.
2. j) Multi-tier: The description fits perfectly the definition of a multi-tier architecture consisting of a presentation tier, an application tier and a data tier.
3. f) Peer-to-Peer: As it is important that a lot of nodes share services and resources and they operate all at the same level, this is peer-to-peer.
4. b) Broker: The description shows that there must be an indirect part between those who request services and those who provide the services – which will be the broker.
5. c) Model-View-Controller: The description is all about separating the data and the visualization of data, which it easy to do through model-view-controller.

Solution Problem 3: Edge-dominant system (5 points)

a) The metropolis model:



The metropolis model represents three communities and is not an architectural diagram:

- Customers and end-users
- Developers
- Prosumers – Users who both produce and consume content

The model identifies different types of stakeholders:

- Outermost does not contribute with content, but provide requirements for the Edge-dominant system
- Middle are prosumers, which is important to support through tools and APIs
- Core: Stable and robust software with collection of APIs available for prosumers.

b) Open source: Users can go from end-user to developer to core developer
Open content: users go from end-user to contributor.

Solution Problem 4: Cloud Architecture (4 points)

a) Software as a Service (SaaS): Consumer uses applications running on a cloud (e.g. Google Docs).
Platform as a Service (PaaS): Provides a development and deployment platform in the cloud (e.g. the LAMP stack, Linux, Apache, MySQL, Python.
Infrastructure as a Service (IaaS): Provides a virtual machine for the developer or system administrator who can run any operating system and software on the virtual machine.

b) Economies of scale (power costs, infrastructure labor, security & reliability, hardware cost), Utilization of equipment (time of day, time of year, resource usage patterns, uncertainty), Multi-tenancy (single application for multiple consumers – save help desk support, upgrade all, single version)

Solution Problem 5: Quality Attribute Scenario (6 points)

a) Quality attribute scenario on usability:

Source:	New user of Spotify
Stimulus:	Learn to use all functionality in Spotify
Environment:	Runtime
Artifacts:	The whole system (Spotify)
Response:	Provide searchable help functionality to help the user discover and find functionality
Response Measure:	Learn to use all functionality in Spotify in average 15 minutes

b) Quality attribute scenario on availability:

Source:	Hard drive
Stimulus:	Crash
Environment:	Normal operation
Artifacts:	Music library database
Response:	Notify system administrator about hard drive failure
Response Measure:	Redundant hard drive will take over with no downtime for user

c) Quality attribute scenario on performance:

Source:	100 000 simultaneous user clients
Stimulus:	Periodic music search request
Environment:	Normal mode
Artifacts:	Spotify music database server
Response:	Respond to the user request
Response Measure:	Average latency 1 second.

Problem 6 Design a software architecture (30 points)

a) Architectural Significant Requirements/Architectural Drivers – 2 points

- ASR1: Modifiability: Should support a variety of user interfaces and functionality configuration including different buttons and screens and interaction through network using smart phone. It should also be easy to update various parts of the system.
- ASR2: Performance: The robot lawnmower must be able to stop or change direction before harming an object.
- ASR3: Availability: The robot lawnmower must be able to operate for weeks without software crash or getting physically stuck.

b) Architectural tactics and patterns – 3 points

Modifiability:

- Increase Cohesion
- Reduce Coupling
- Generalization
- Model-View Controller
- Elfes Reference Architecture (Layered)
- State pattern

Performance:

- Prioritize events
- Reduce overhead
- Schedule resources

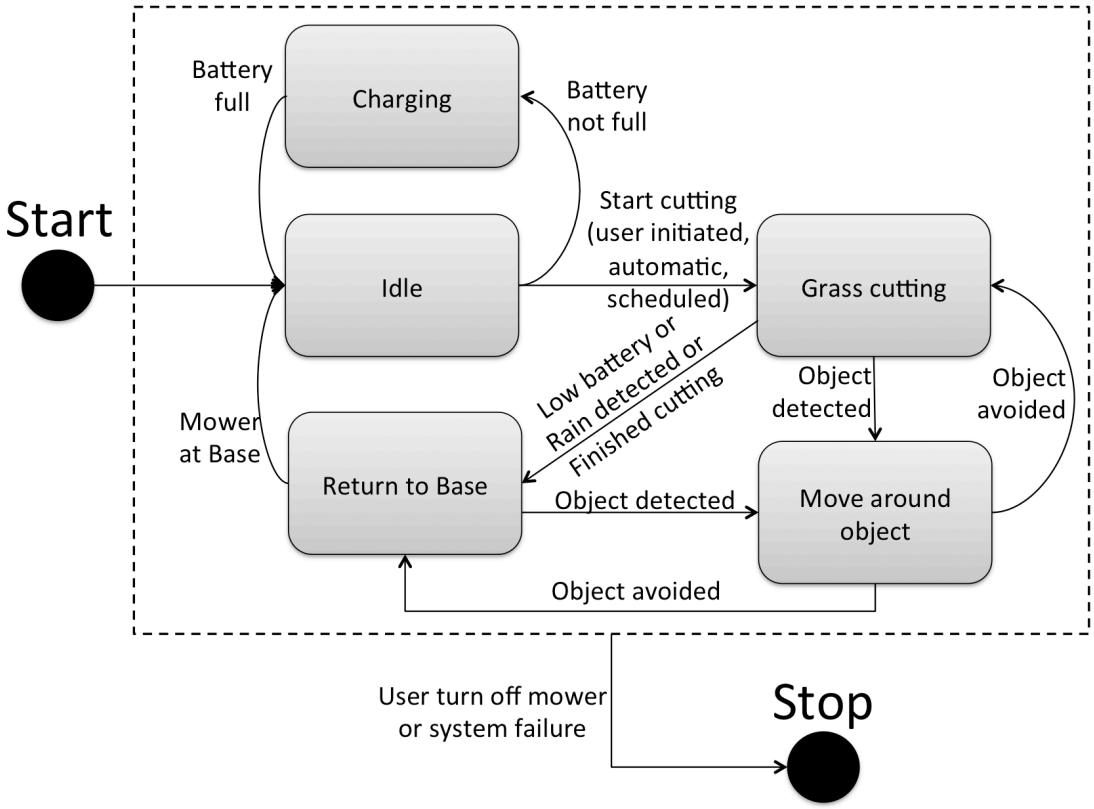
Availability:

- Self-Test
- Exception Handling
- Heartbeat

c) Process view – 8 points

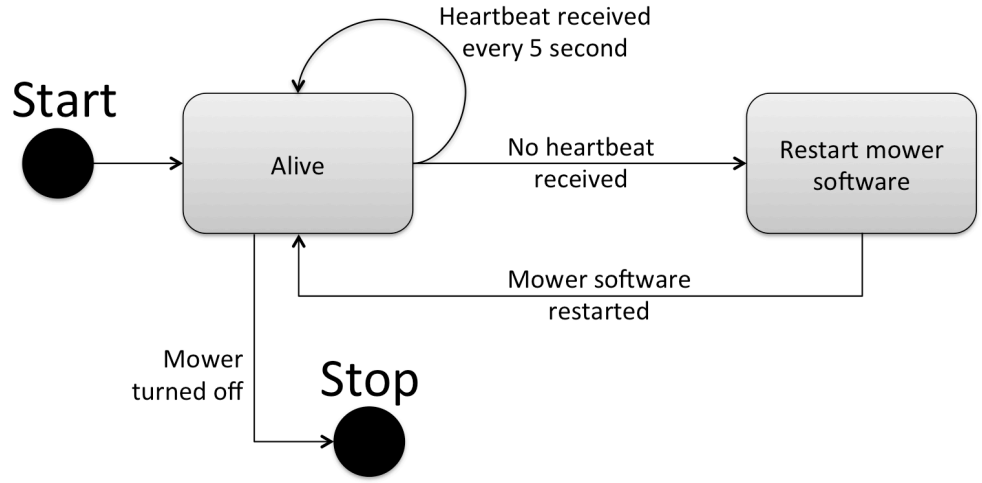
The process view shows the states of the software running the mower, as well as the software that runs the heartbeat that checks if the mower system software is running.

The state diagram for the mower system software can be the following:

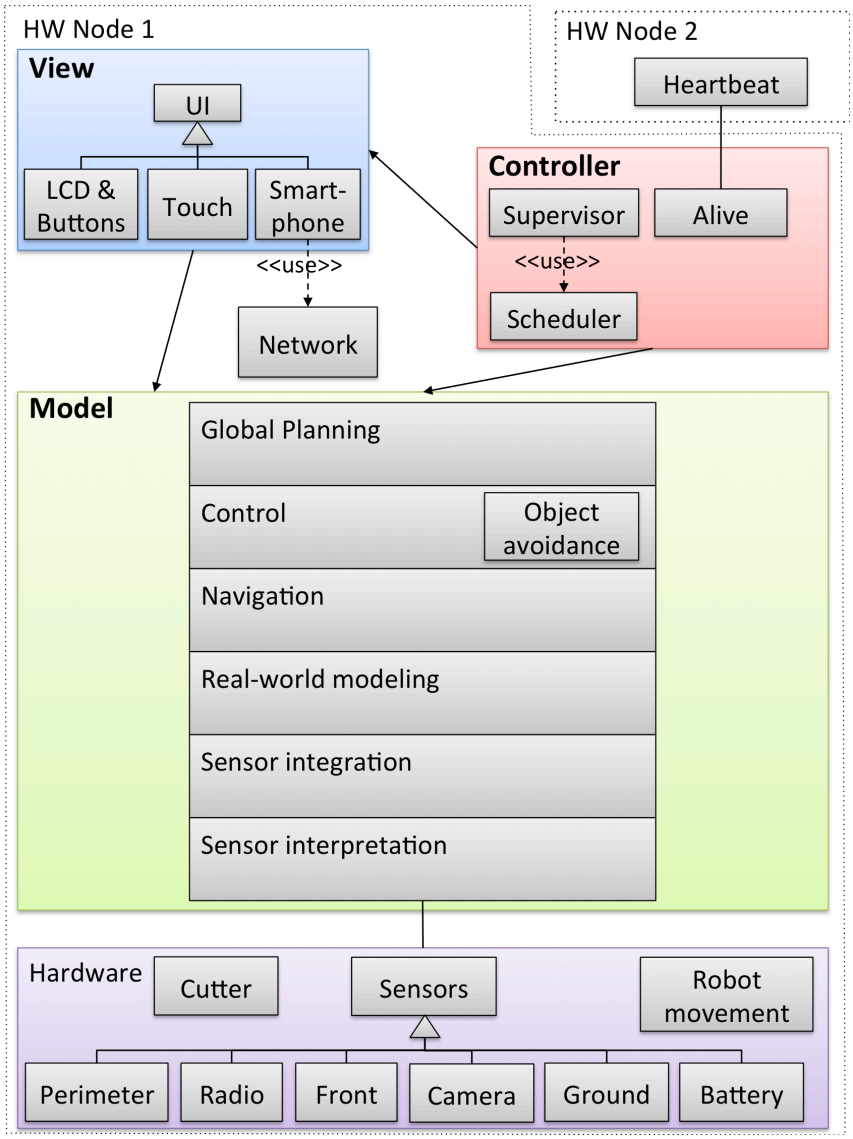


After starting up the mower, the system will be in an *Idle* state. If the battery is not full, it will go to *Charging* state until the battery is fully charged. The mower system software will go from the *Idle* to *Gras cutting* state if the user will initiate it using the user interface on the mower, or it will be initiated by the software if the user has chosen the automatic mode or schedule mode. The mower will go from *Gras cutting* state to *Move around object* state if an object to be avoided is detected, and will return to *Gras cutting* state after objects has been avoided. If the battery is low, it has started to rain or the mower is fished cutting the grass, it will go to *Return to Base* state where the lawnmower will move back to Base, and avoid objects if detected (*Move around object* state) on its way back. When the mower has returned back to its Base, it will change to *Idle* state.
If the user turn off the mower or the system fails, it will shut down the system.

The second state diagram shows the lifecycle of the heartbeat mechanism:



d) Logical view – 14 points



The logical view is design by combining two main architectural patterns: The model view controller and Elfes Reference Architecture, which is a layered architectural pattern. The first layer is the hardware abstraction layer, which deals with interfaces to the physical hardware related to sensors and activators. Cutter is a motor for adjusting the height of the grass cutter, and robot movement involves the motors and activators for moving the robot around. The hardware layer is kept outside the MVC, as it is not related directly to none of them.

The Sensor interpretation layer will handle the input for the sensors and produce raw data. The Sensor integration layer will combine sensor raw data (e.g. over time, remove noise), to get higher quality sensor data at a higher abstraction level. The Real-world modeling layer models the surround world of the robot using the data from the sensor integration layer. The Navigation layer manages basic local navigation using the world model from the layer below. The Control layer makes a plan for local navigation based on data from Real-world modeling. This layer also contains an Object avoidance class which can avoid detected objects which should be avoided. The Object avoidance class short-cuts the two top layers, to make it more

efficient. This means that the Object avoidance class can initiate movements of the mower without going through the Global Planning and the Supervisor.
The Global Planning layer is responsible for high level scheduling and planning of the robot's actions.
The final layer in the Elfes Reference Architecture (Supervisor) is placed in the controller. The Supervisor is responsible for overall decisions on what the robot should do according to the mode it operates in (manual, scheduled, or automatic), and the state it is in (Idle, Charging, Grass cutting, Move around objects or Return to base). The Supervisor uses the State design pattern. The two other classes in the controller are the Scheduler (which keeps track of time (to start cutting grass at time intervals), and Alive which will send a message to another hardware unit every 5 second. Alive can also force a re-start of the system software.
The View consists of variations of UI through a generalization: LCD & buttons, Touch and Smart-phone. The Smart-phone interface uses a Network component< to communication with the smart phone over Wi-Fi or 3G/4G.

e) Architectural rationale – 3 points

The emphasis of the design is on modifiability, availability and performance. The Elfes Reference Architecture (layered) and Model-View controller patterns were combined to allow maximum flexibility in terms of modifiability. The performance issue for the mower was to ensure to avoid objects in time. This was solved by short-cutting the two top-layers if an object which had to be avoided was detected. Availability has been solved using heartbeat to a separate hardware unit, which will initiate a hardware reset if no heartbeat is received within 5 seconds.