Software Architecture Report

Team: FT_3
Version: 1.4
Authors:

- Bart de Man
- Felix Barten
- Joel James Bartholomew
- Spiros Tzavaras

Table of Contents

Stakeholders

In order to find the necessary requirements and functionalities we have composed a list of stakeholders in our project. We have decided to split the stakeholders into small groups for better traceability of requirements. All the internal company stakeholders have the prefix QC.

Our stakeholders for this project are the following:

(Less-abled) Product users

The end users of our product are people who are physically or mentally less able to perform household tasks. The goal of our product is to improve the Quality of Life of our customers. People who can still live at home instead of a full care nursing facility but have difficulties with performing certain (household) tasks. Our product should be able to alleviate the amount of work a resident has to perform or even reduce the amount of visits required per week from outside help. The concerns of the product user for this product are:

- Usability
- Reliability
- It should be easy for them to learn robot's functionalities
- Availability
- The robot should be secured from outside influences.
- The robot should provide security for sensitive data stored on it
- The robot must not hurt the product user.
- The user would like the robot to be a durable product. The following attributes are expected:
    - Water resistant
    - Shock resistant (falling)

Relative or family member

A relative or family member is a stakeholder in our project because they are likely responsible for configuring the robot (when the user receives one most likely). These relatives should be able to instruct the robot (See Logical view services) without much prior knowledge and should be able to customize the robot's settings, such as how the user wants their coffee.

- Usability
- Modifiability of behaviour

QC: Quality Assurance Team

Quality assurance will be involved with the whole development of the product to ensure the product adheres to the requirements, to test and improve the entire development process.

- Testability
- Well-defined requirements
- Making a better product

QC: Software Testers

This group of stakeholders might have insights to make sure the software of the product is and remains testable.

- Clear details about QC-Robot's functionality and design should be provided
- Testability (for the whole product and for the separate modules)
- Interoperability with other systems

QC: Hardware Testers

This group of stakeholders might have insights to make sure the product is and remains testable.

.

- Clear details about QC-Robot's functionality and design should be provided
- Testability

QC: Developers

The developers will be responsible for creating the software behind the robot. The developers are a stakeholder for this project because they would like a well-maintainable software architecture which can be maintained and modified in the future without major overhauls.

- Interoperability with other systems

- Platform compatibility
- Memory utilization
- Modifiability
- Maintainability

QC: Researchers

They will be conducting the research that will clarify uncertainties surrounding the product.

- Clearly specified requirements
- Subjects for research

QC: Hardware Specialists

They will be responsible for the hardware designs needed by the product

- Clear specifications about the functions hardware should be able to perform

QC: Managers

Management wants a profitable product and to preserve the continuity of the company.

- Profitable product
- Better than similar products
- Preserve company’s status

QC: Investors

Investors are the reason our company is able to conduct research and build products. As Quality care is a small company we need a lot of outside funds to get our company off the ground. Investors have a large stake in our first commercial product as it will indicate if they will be rewarded for their investment in our company. The main concerns of the investors are:

- Schedule and budget estimation
- Requirements should be properly implemented

QC: Software Architects (FT_3)

The architects want a well-functioning system because they are responsible for the architecture

- Requirements traceability
- Consistency of architecture (parts should be combined successfully)

Doctor

They will administer the medicine required by the patient and track their health. They will be able to administer medicine and adjusting or adding prescriptions from their office using a web interface. Less need for the less-abled to visit, thus saving time and money.

- Usability

QC: Software maintainers

They will be maintaining the software for the product

- Interoperability with other systems
- Platform compatibility
- Memory utilization
- Maintainable system
- Modifiability
- Reusability

QC: Service engineers

Service engineers are a stakeholder in this project as they will be responsible for servicing the robots after launch of the project. The service engineers would like the robot to have easy to replace parts so the replacement of certain components is as easy as possible. The service engineer can use his/her experience to influence the development to make hardware that is easy to service. The service engineers' main interests are:

- Remote analytics
- Modular Hardware
- Easy (dis)assembly of the product
- Clear maintenance instructions

QC: Customer Support

They will get called when any users run into difficulty using the product

- Well-documented instructions for troubleshooting
- Clear documentation of QC-Robot

Health Insurance Providers

Health insurance is a main stakeholder for our product. Health insurance companies may be able to save money if the product is cheaper (and thus saves them money) than care provided by humans. This will allow medical personnel to perform less mundane tasks and focus on other aspects of care.

Health insurance companies might subsidize robots if they want people to transition to other types of care. This method would incentivize end users to buy the product if a part is refunded. Or alternatively the insurance company may offer the robot as a substitution of human care. Because of the position of the health insurance companies these stakeholders will be viewed as customers in this case.

- A robot is cost effective alternative to real personnel and they would like to make more accessible for users in need of help
- Availability
- Security
- Robot should be safe for public use (see Regulatory Authority)
- Robot should not harm patients
- Licensing

External Manufacturer for the Hardware

The manufacturer has to make sure the product can be produced once the designs are finalized.

- Clear specifications on the hardware needed for the product
- The design provided by the designers at QC should be realistic and be able to be manufactured by the designated hardware manufacturer

Regulatory Authority (Nederlandse Zorgautoriteit(NZa))

The NZa is a stakeholder in this project because the product will be used in a medical fashion and can have effects on public health if it is not deemed safe. To make sure the product will comply with the latest rules and regulations the NZa is a definite stakeholder in this project.

- Product should be safe for use and should comply with all health care regulations
- Security

Suppliers/Distributor/Trader/Retailer/Wholesaler

The distributor will deliver the product to retail stores or directly to the customers

- Distributors want the product to be easily moveable without any special ways of transportation required

Third party developer

Third party software developers will add additional behaviour to the robot's default behaviour. With the help of third party developers the robot could provide extra methods of care or more social interaction with main users of the system (See logical view). Third party development is also a good way of gaining additional exposure for the product if a lot of interesting new functionalities are developed. However, these additions need to be curated otherwise the product will not be safe for the public anymore as external developers might not adhere to the health regulations which the product's programming is bound by.

- Modifiability of the code or rather good interfaces/API's to interact and extend the existing system.
- Interoperability with other systems

Future Stakeholders

Third party addition curator

The third party curator is a future stakeholder of our project. The 3rd party curator's job will entail checking packages to ensure the public's safety. This stakeholder is not listed in the

current stakeholders because it is dependent on the adoption rate of the project. If the project is widely adopted after being completed it is likely more third party development will occur which will lead to more packages that need curation.

- Modifiability
- Testability
- Supportability

Requirements

After deciding on the list of stakeholders we have created a list of requirements in order to give our architecture more meaning. These requirements are split in functional and nonfunctional requirements. These requirements were used to formulate the main functionalities.

Functional requirements

- Users should be able to customize how they like their meals
- Users should be able to customize the reminder settings
- It must be able to open doors and receive guests
- Unit must be able to notify the user that it is in need of servicing
- Unit must be able to analyze faults
- Unit must be able to send diagnostics back to service engineers for analysis
- Unit must have modular hardware
- Unit must be easily disassembled
- Unit must have a (maintenance) instruction manual
- Unit must have facial recognition
- Unit must be able to recognize voice commands
- Unit must be able to recognize hand gestures (sign language) commands
- Unit must be able to be controlled remotely via a remote.
- Unit must be able to be configured
- Unit must have a display to be able to accommodate deaf or near deaf people
- Unit must be able to respond to events 0-10 seconds
- Unit must be able to move around at a reasonable pace
- Unit must have a simple user interface
- Unit’s behaviour can be modified from extra software defined by third parties
- Unit must be able to choose what the next task is that it will perform (scheduler)
- Unit must be able to show its status (either by voice or a graphical representation)

- Unit must be able to send and receive secure video from and to QC headquarters
- Unit must be able to store medical information about the user.
- Unit must be able to detect heart rate of the user
- Unit must be able to measure blood pressure of the user
- Unit must be able to take the user's temperature
- Unit must have a bluetooth connection
- Unit must have a WiFi connection
- Unit must have a microphone
- Unit must have motion detection sensors
- Unit must have an inventory of available medicine
- Unit must be able to order additional medicine if the medicine runs out.
- QC Control Centre employees must be able to view a secure video from the robot
- The product must be able to monitor health information such as heartbeat and blood pressure

Non-Functional requirements

- Unit must comply with health and safety regulations
- A detailed description of unit's functionalities
- Unit must be water resistant
- Unit must be shock resistant
- Unit must be assembled from components that are able to be tested individually
- Unit must have replaceable parts
- Parts of the unit must be able to be recycled
- There must be documentation to give information about the robot
- User must be able to get acquainted with the basic robot functionalities in less than 15 minutes
- Unit's power consumption must be minimized where possible

Main Functionalities

After documenting the stakeholders of this project and its requirements, it was decided to create a list of functionalities for the robot. These functionalities are focussed on the end user and should all raise the quality of life for the user of this product.

1. Must be able to help the resident get back up if he/she has fallen down
2. Must be able to remind users of (important) subjects/appointments

3. Must be able to respond to a doorbell if the resident/end-user may not be physically able to respond to the door bell
4. Must be able to make meals and/or drinks available. The resident may not be able to cook for him/herself
5. Must be able to clean the house using a built-in vacuum, thus removing the need for (extra) personnel to do so.
6. Must be able to administer medicine to the resident and keep the medicine stocked via prescriptions.
7. Your QC Robot should show (secureline) video in the QC control centre located in the (nearby) area.

Referenced Architectures

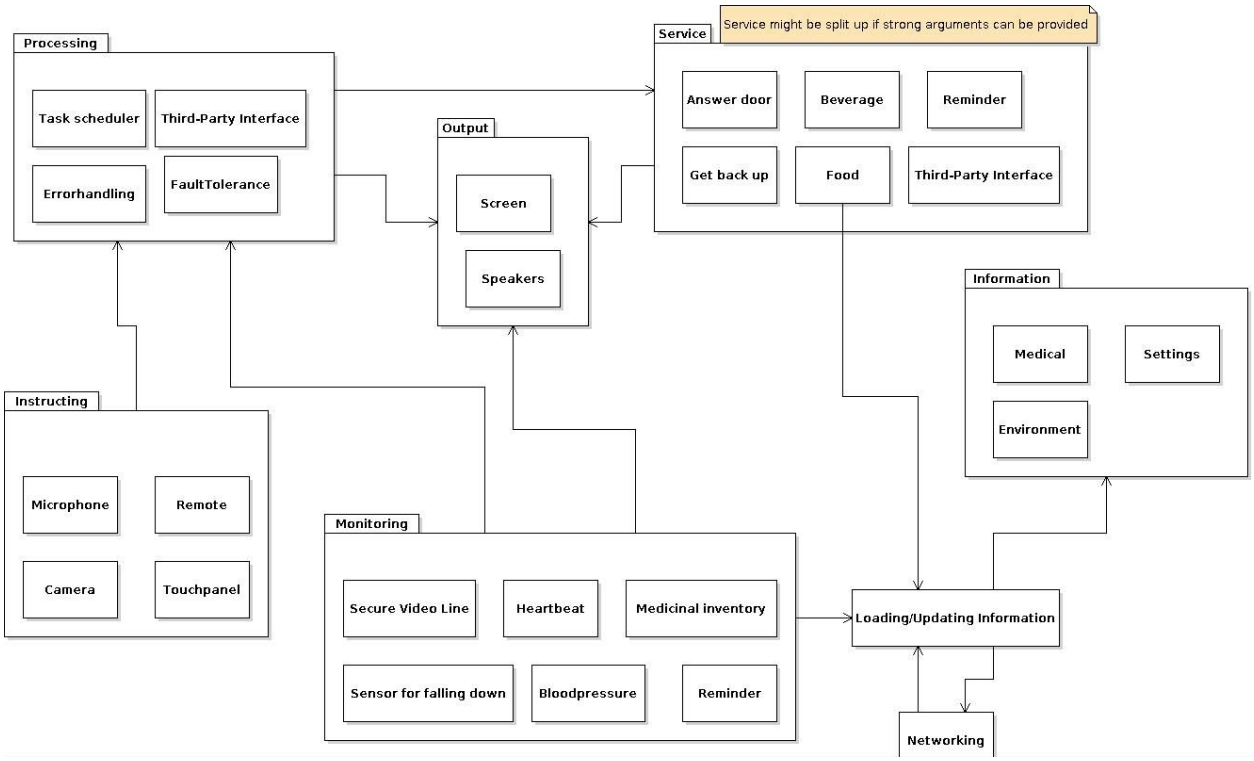
- JAUS reference architecture: uses a message passing protocol to provide interoperability among subsystems and components that compose systems resulting from this architecture
- ACROSET reference architecture: ACROSET is a component-oriented reference architecture for tele operated service robots and its main characteristic is the reuse of components from different systems. This architecture is composed by subsystems such as UIS (User Interaction Subsystem)
- Servicebots reference architecture: it was designed to service robots in indoor environments and is composed by 3 subsystems, service bots, fix bots and softbots. The type service bots is a robot capable of driving autonomously in an average complex environment only with sensorial information. The fix bots present sensor and actuators distributed all over the environment, having their own intelligence, whereas the softbots refers to the software agents executing various tasks for the requesting user
- Robot tele operation reference architecture: its main identified components are graphical representation, collision detection , user interface, communications and controller.

Provided in **appendix A**, some referenced architectures in detail.

4+1 Architectural View Model

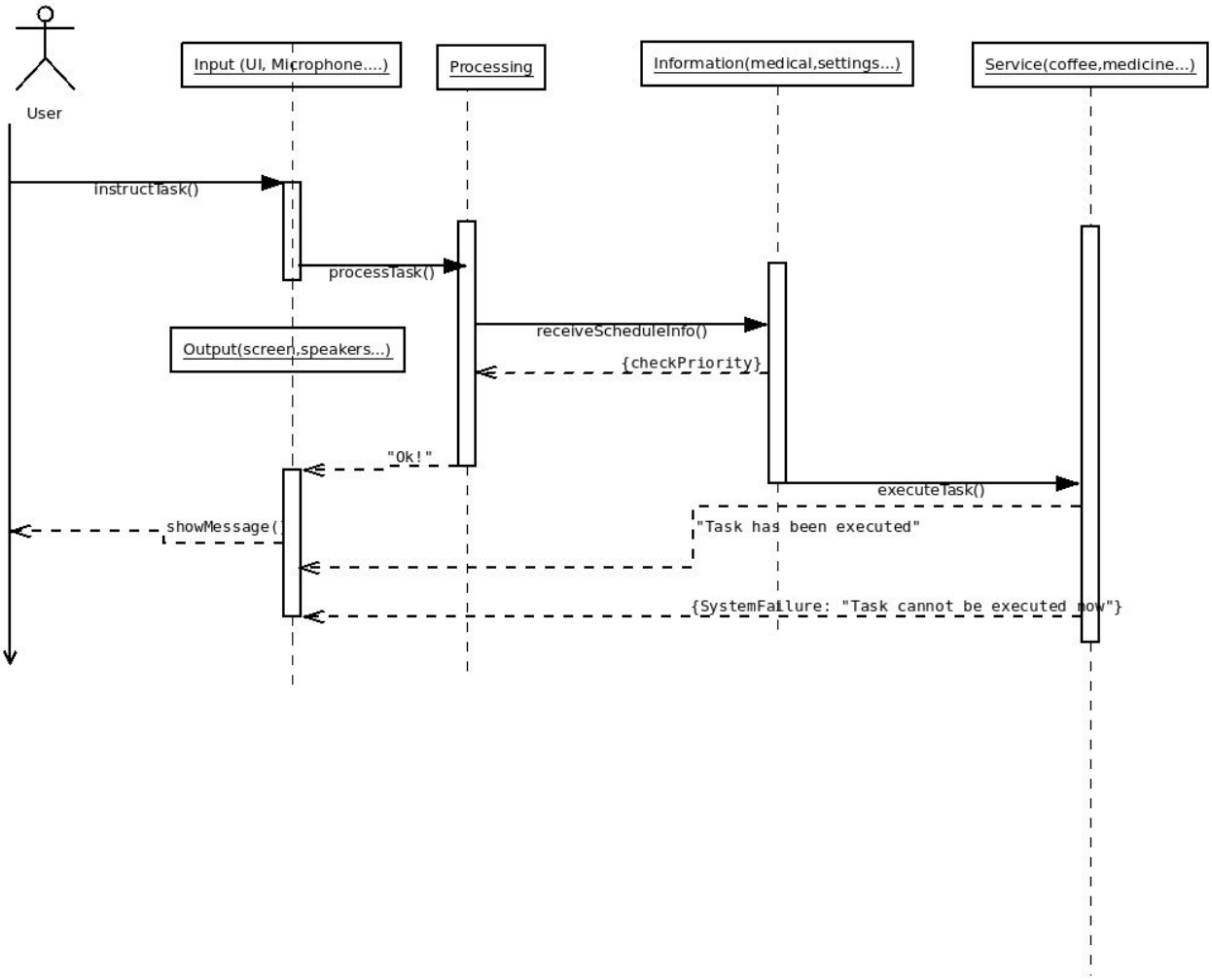
Logical view

Below a class diagram is provided, that represents the functionalities that QC-Robot provides to the system. The logical view has been designed in such a way that the end user can understand the workings of the robot.



Process view

As for the process view, a sequence diagram has been created, in order to show how QC-Robot handles events, and how each of the components acts in a process.In case of a critical failure, which might occur in any time while invoking one of the components, a relevant message is shown to the user.


User
Input (UI, Microphone....)
Processing
Information(medical,settings...)
Service(coffee,medicine...)
instructTask()
processTask()
receiveScheduleInfo()
Output(screen,speakers...)
{checkPriority}
"Ok!"
executeTask()
showMessage()
"Task has been executed"
{SystemFailure: "Task cannot be executed now"}

Deployment view

The diagram describes the software components of the system, as well as the way these are connected to each other.

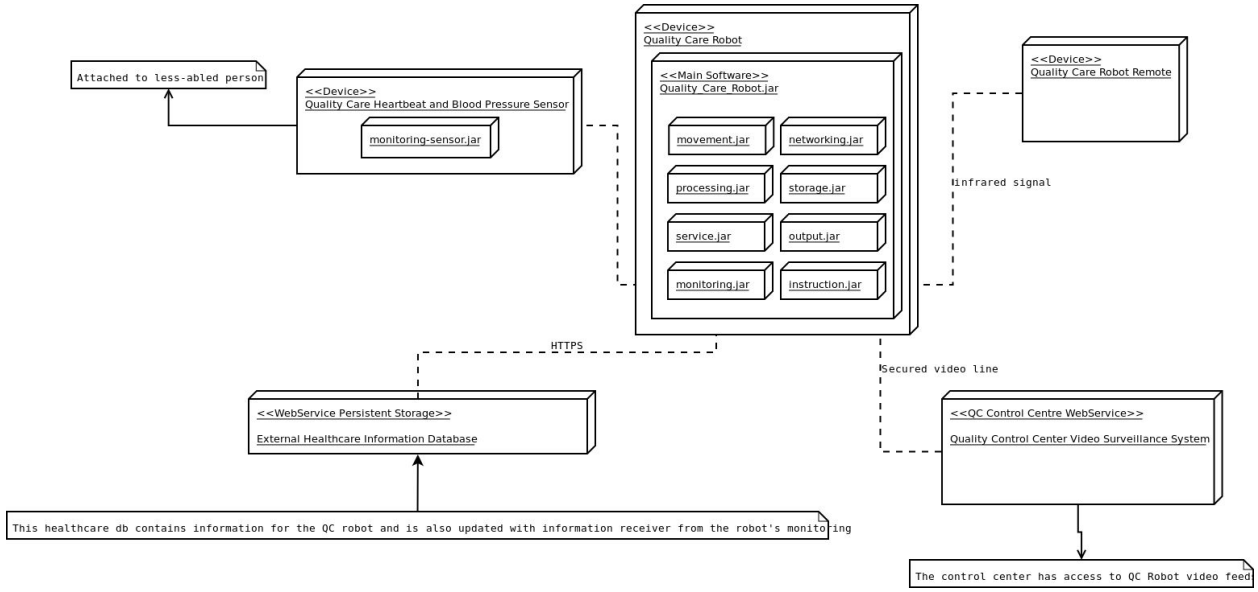
The deployment has the following main elements:

1. A Quality Care sensor for monitoring heartbeat and blood pressure
2. A remote control for the Quality Care robot
3. The Quality Care robot itself
4. A web-service which allows the robot to make its video feed available to the Quality Care control centre

There is also a service which provides access to the health information of the primary user of the robot.

The robot has the following software components:

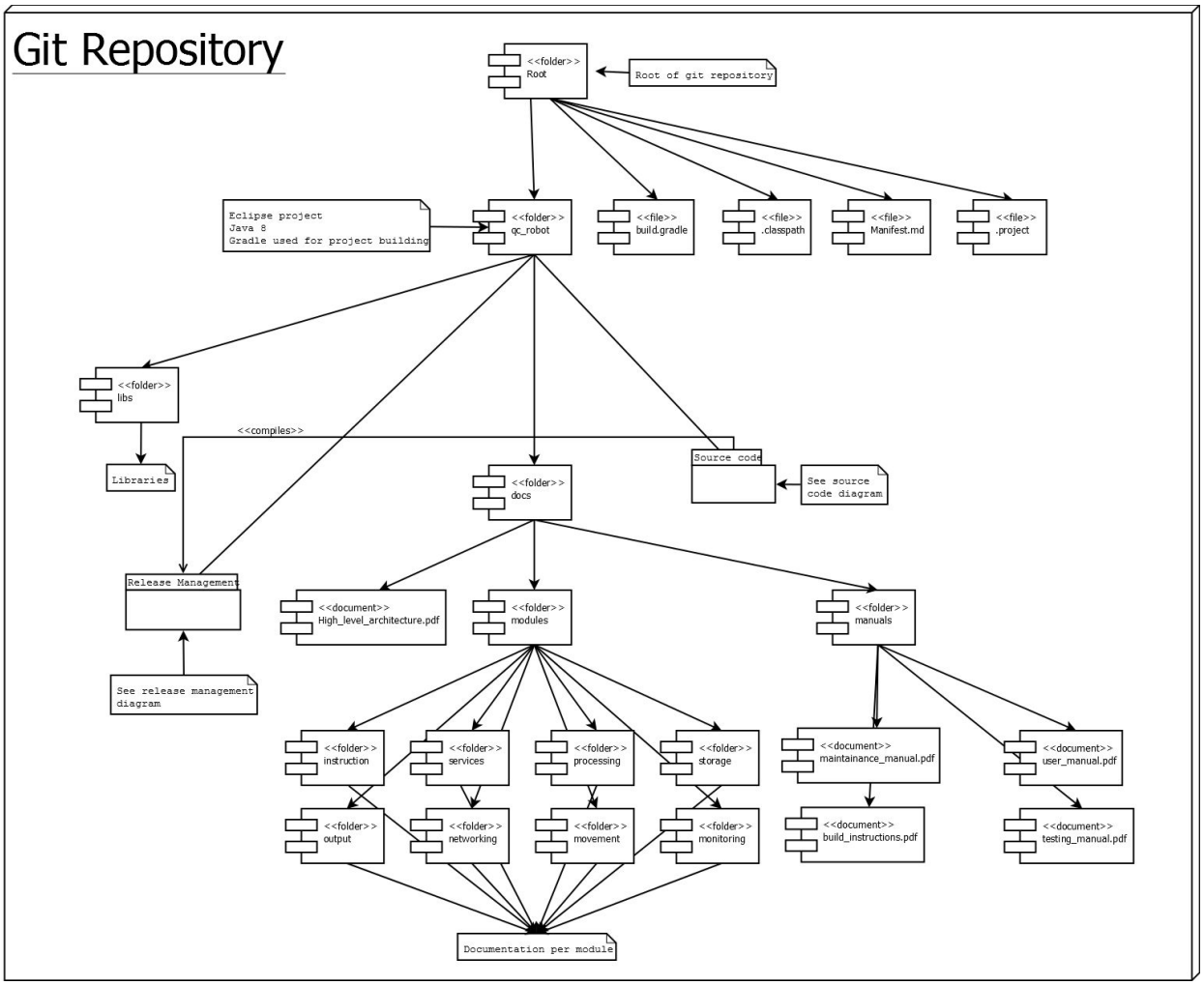
1. Instruction Module - The user interface which has several sensors which the user can use to issue commands. Also the robot will react to detected signals of the sensors to autonomously execute actions.
2. Processing Module - This module validates the input provided by the instruction module and calls the service corresponding to the command issued
3. Service Module - Contains the main functionalities of the robot.
4. Local information Storage - Contains information relevant to the robot’s settings and user information
5. Monitoring Module - Used to monitor the product user’s health metrics
6. Networking Module - Allows the robot communicate over network connections
7. Movement Module - Used by other modules to allow the robot to move
8. Output Module - Used to provide feedback (information) to the user via the speakers and the display screen.



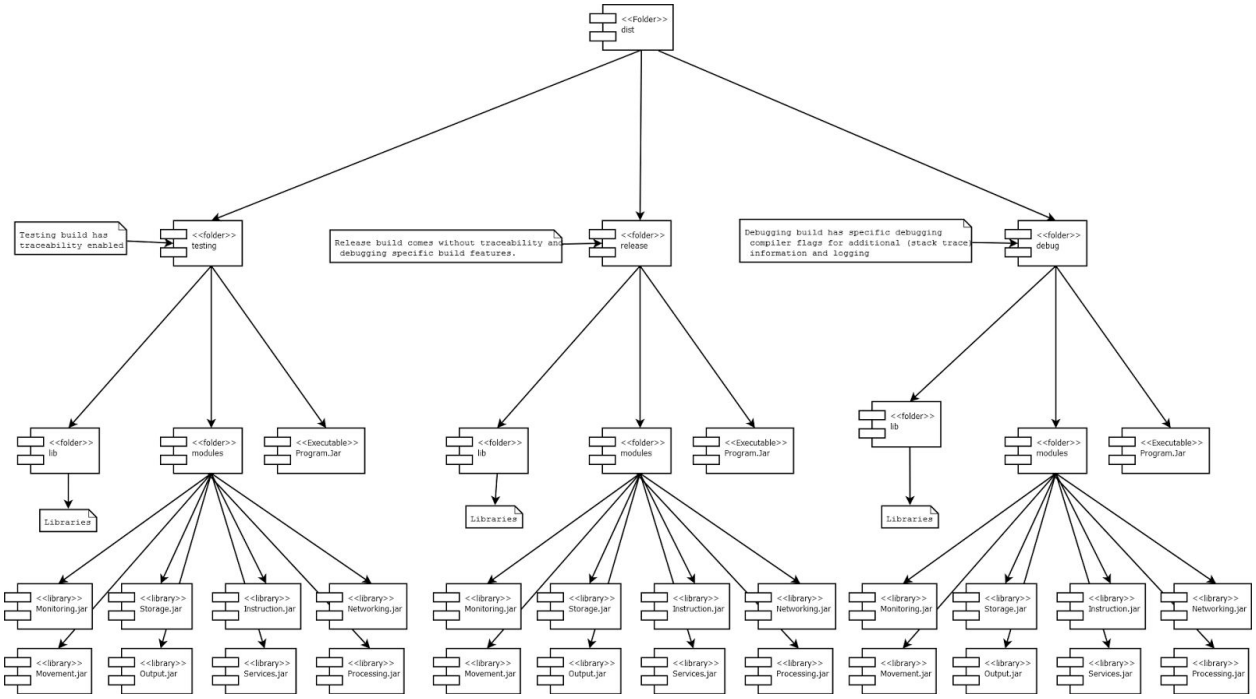
View image on github

Implementation view

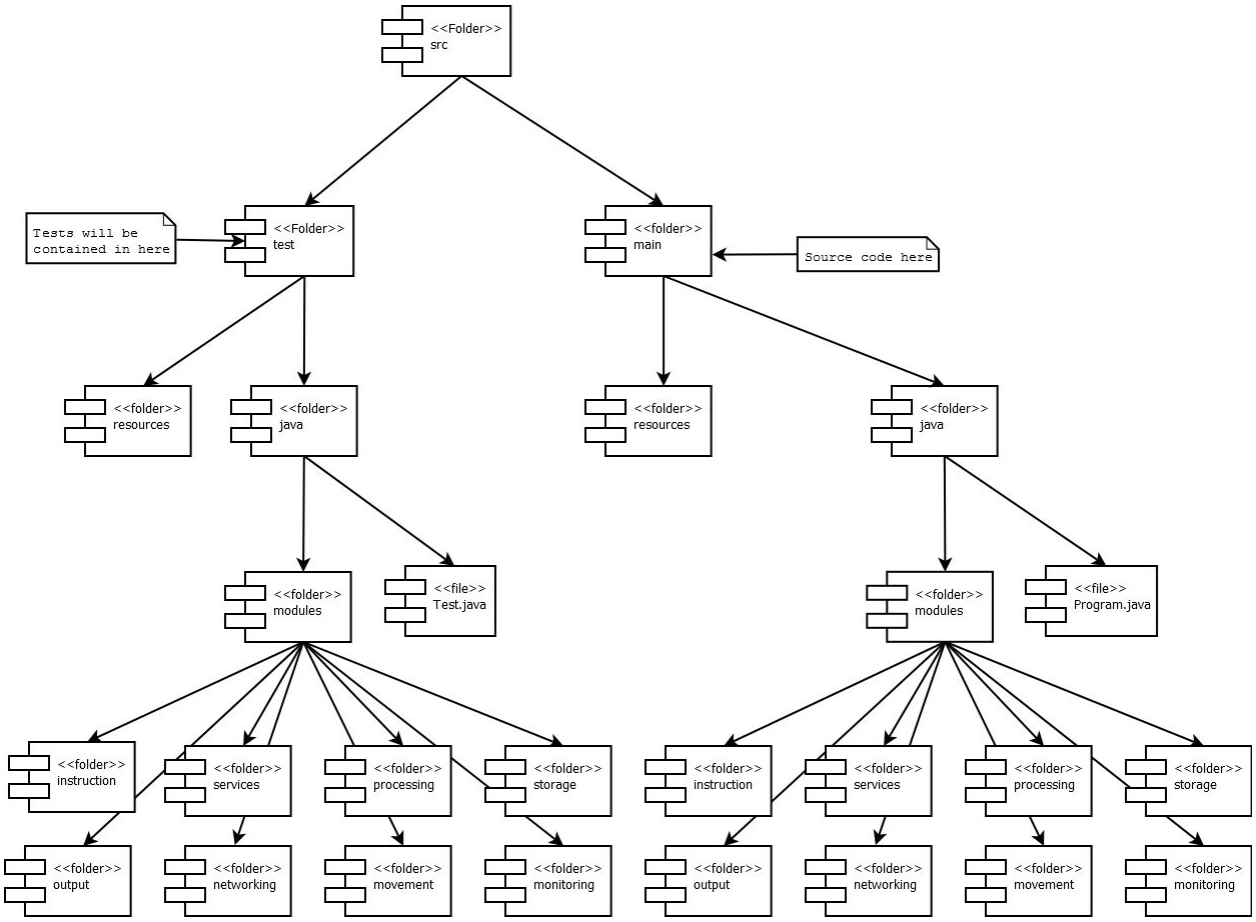
The structure of the code, the directories and files of the system, the environment and in general everything that concerns the programmers of the system, is represented in the following diagram.



Implementation overview (above)



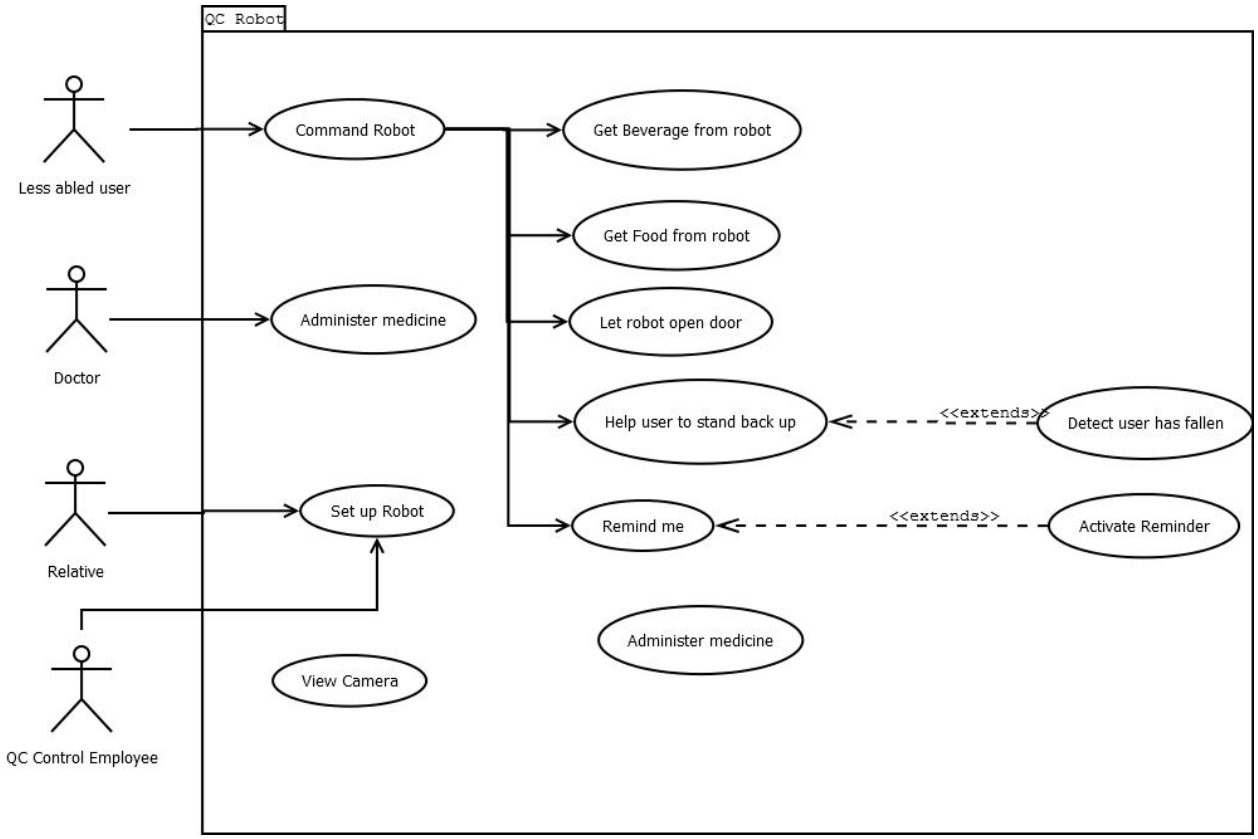
Release management diagram (above)



Source folder diagram (above)

Use case view

Below we can see the interactions between the main stakeholders and the processes of QC-Robot. Two of the use-cases below are attached with an automated process which checks if actions are required.



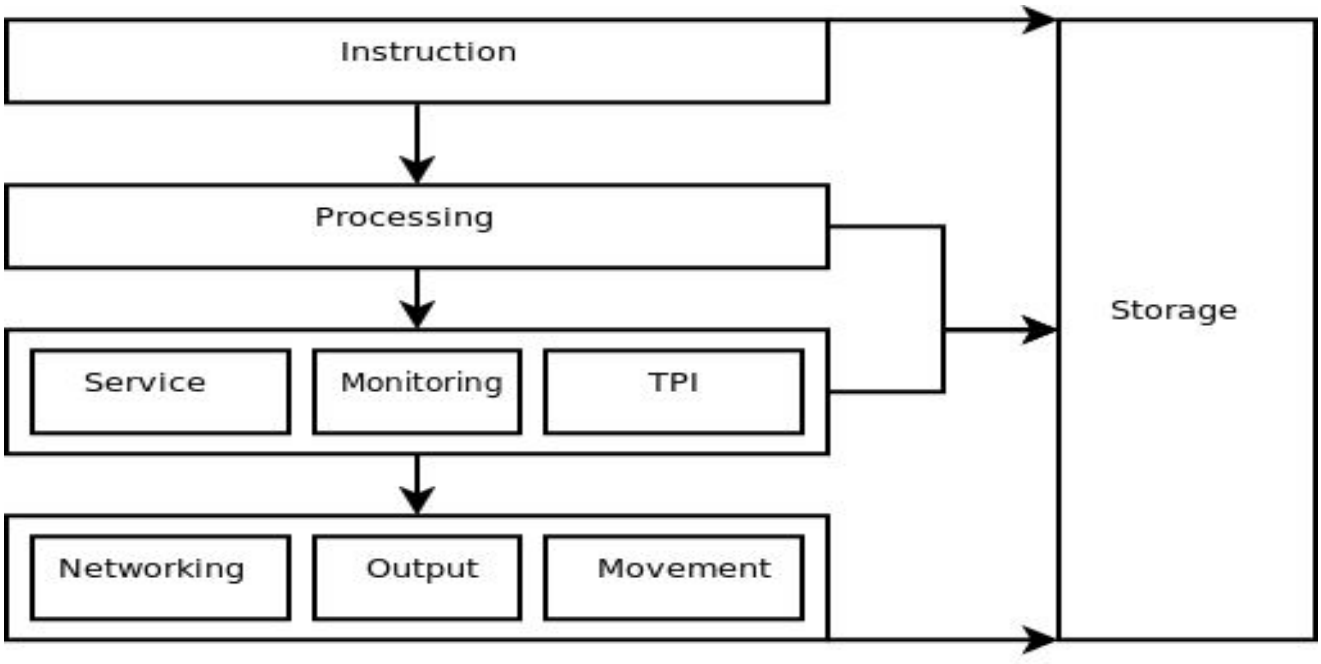
Architecture patterns

Except for the pipe-filter pattern, we have studied a couple of architecture patterns that may or may not be useful in the creation of QC-Robot's architecture. By listing these patterns and weighing in their pro's and con's, we can make the right decision in order to apply the right pattern(s). First, we mention some patterns that we believe could fit QC-Robot's architecture:

1. Client-Server pattern.Even though server side could affect system's performance in a really bad way, this pattern could easily be used to allow the QC control centre to access

the video feed of a QC robot. Each robot will be able to communicate with the QC control center to make its video feed available.

2. Layer pattern. We decided to incorporate a layer architecture in our system to allow the different modules to be developed independently and to ensure that the system will be able to adapt to any future changes. Modules offer a cohesive set of services, where each layer can use any other layer that is below it, on the same level as it or to the right of it. The structure of this pattern is shown below.



Now, we mention some patterns that we do not believe would appropriately fit QC-Robot's architecture.

1. Peer-to-peer pattern. There is no need for the QC-Robot to communicate with the control centre in a peer to peer fashion. The client server architecture is a much better solution
2. Model - View - Controller pattern. Although this pattern could provide our system with certain advantages, we believe that complexity may not be worth it, because we want

the User Interface to be as simple as possible, as mentioned in the system's requirements.

3. Multi-tier pattern . Although this pattern could somehow fit QC-Robot's architecture in order to fulfil some requirements of the system, such as its reusability, it is more suited to components running on different systems and so layering pattern would be preferable.
4. Broker pattern. This pattern would not fit for our robot's software as the robot's services would all be contained within it's own software and not on external web servers. The broker pattern creates an intermediary object to interact with these external services and creates an abstraction between itself and the other layers because those layers don't know which service they're communicating with. However this pattern would simply not apply to our current architecture.

Quality Attributes

For our first part we have decided to focus on the security of the storage part of our system. The storage part of the robot is responsible for saving and reading various types of information. Among these types of information some parts are really privacy sensitive and for other types of information a less secure type of storage can be utilized.

Design decisions

For documenting our storage decisions we have used the template proposed in the paper written by Tyree and Akerman[1]. Our architecture decisions in week 4 are centered around the storage part of the robot. As our robot deals with different types of information (medical, environment and settings) we have chosen to focus on the medicinal information first as this type of information storage poses the largest architectural challenge out of the three types.

Storage decision #1

Issue	The storage of health information must comply with certain security standards/constraints specified by regulatory administrations. (CBP: https://cbpweb.nl/en, NVZa:"http://nvza.nl/", FDA:"https://nl.wikipedia.org/wiki/Food_and_Drug_Administration")
Decision	Work will begin on the storage mechanism as soon as possible so that the auditing of the it should have minimal impact on the release date of the product. The encryption used should be easily modifiable in the case of it being incorrect or for if standards were to change in

	the future.
Status	Pending
Group	Storage
Assumptions	● The information will be stored on a piece of hardware attached to the robot. ● The exact cost cannot be specified but it must adhere to health care regulations regardless of the cost. ● It takes time for the regulatory administration to audit the product so this module should be done a month before the robot's completion. ● There will be data encryption ● There are no storage technology standards ● There are no encryption technology standards. ● Due to possible changes the applied encryption standards it should be possible to swap out encryption implementations.
Constraints	The health information storage must be audited before the product can be released.There are regulations concerning this type of data storage.
Positions	Before the storage mechanism will be built all information on the regulations and standards must have been received from the various administrations. The system will only adhere to the those provided. Why not: In the ideal situation this will be cheaper and take less time but may end up being much more expensive in the worst case scenario
Argument	Due to the fact that product must be audited by another entity and there is less control on that aspect the storage mechanism of the robot should be submitted as soon as possible for auditing and it should be flexible enough to comply to changing expectations.
Implications	It has an effect on the planning of the building of the storage mechanism and the release of the product. It will cost more effort and time than the alternative. The implementations of mechanisms concerning encryption and security must be easily changed in case of the product failing the audit or if regulations were to change.
Relation Decisions	The choice of encryption (storage decision 2)
Related Requirements	The storage of medical information
Related Artifacts	Hardware and software storage mechanisms
Related Principles	-

Notes	Architecture Tactics: Limit access, Limit exposure,

Storage decision #2

Issue	A security measure needs to be taken to prevent third parties to be able to access medical information on the robot
Decision	Using (HIPAA) compliant encryption mechanisms to encrypt the data stored on the robot.
Status	Approved
Group	Storage
Assumptions	● It is assumed there are rules and regulations regarding the storage of medical data. These rules and regulations are determined by the regulatory agencies. ● Regulatory agencies require encryption of medical data whenever it is stored. ● Due to privacy concerns relating to the data it is assumed the encryption algorithm that is prescribed by the regulatory agency will have a negative performance impact.
Constraints	If the decryption key is ever lost the information would become irrecoverable
Positions	The data could also be stored elsewhere (not on the robot). This would shift the priorities of securing hardware on the robot to securing a service in a data center and thusly securing the data center itself as the data should be protected. Why not: the approach above would work for our robot as it would save costs developing mechanisms for security and hardware but simultaneously would cost a lot more in data center and (physical) security where it might not be worth it to invest in this alternative.
Argument	Personal data such as health information is of big importance and so preventing accessibility from others is necessary.
Implications	Using (strong) encryption usually results in a lowered performance due to encrypting and decrypting the data while using the storage device.
Relation Decisions	-
Related Requirements	The storage of medical information.
Related Artifacts	Hardware and software storage mechanisms
Related Principles	This decision affects the Performance, Reliability and Security of the

	system. CIA
Notes	Architecture Tactics: Resist attack: Encrypt data, Limit access, Limit exposure

Storage decision #3

Issue	How to restrict access to the storage and who should be authorized actors?
Decision	The only authorized actor should be the doctor. He is the only one responsible for tracking user's health status, and updating some information if needed.
Status	Pending
Group	Storage
Assumptions	● The stored information must be able to be permanently (destroyed/erased) somehow. ● There will be data encryption ● The data should be readable (accessible) and writable.. ● A service engineer should be able to replace the storage unit in case of damage but not have access to the actual data.
Constraints	Most of the CRUD operations must be automatic. This means that, there should be some algorithms implemented, that check user's health status. (i.e. calculating average blood pressure for last days, and updating the dosage if necessary)
Positions	An alternative way would be that more than one of our stakeholders would have access to the information. Why not: Although, this could be a positive aspect for the user, (i.e. if he really doesn't feel well he would be able to take one pill more than he should), but it is a fact that such actions should be approved by the doctor first.
Argument	This way, there is no need for actors apart from doctor to make CRUD operations on database
Implications	Doctor is able to use a simple UI to collect/update data. User and relatives can be informed about user's health status from the doctor.
Relation Decisions	The choice of encryption (storage decision 2)

Related Requirements	The storage of medical information
Related Artifacts	Hardware and software storage mechanisms
Related Principles	Confidentiality, integrity and availability (CIA)
Notes	Architecture Tactics: Resist attack: Iden/Auth/Auth actors

Storage decision #4

Issue	Should the system use only software or hardware level security?
Decision	Usage of hardware security and software security.
Status	Pending
Group	Storage
Assumptions	The system must be secured on at least one level
Constraints	-
Positions	Secure the health information storage on a hardware level so that it is only possible for someone specialized from Quality Care to remove it. Why not : See below Secure the health information storage on a software level so that only authorized individuals can access it. Why not: Having redundancy by combining more actions will lead to a more reliable security system due to their not just being one point of failure.
Argument	A combination of hardware and software level security is more effective than just one or the other. The two levels will complement each other and if one is comprised it does not necessary mean that the system is breached. Using an encryption protocol will ensure data security if it is properly implemented.
Implications	It will have a positive effect on the security of the system.
Relation Decisions	-
Related Requirements	The storage of medical information
Related Artifacts	-

Related Principles	Confidentiality, integrity and availability (CIA)
Notes	Architecture Tactics: Detect Intrusion, Separate entities, Inform actors (in case of attack), Revoke access

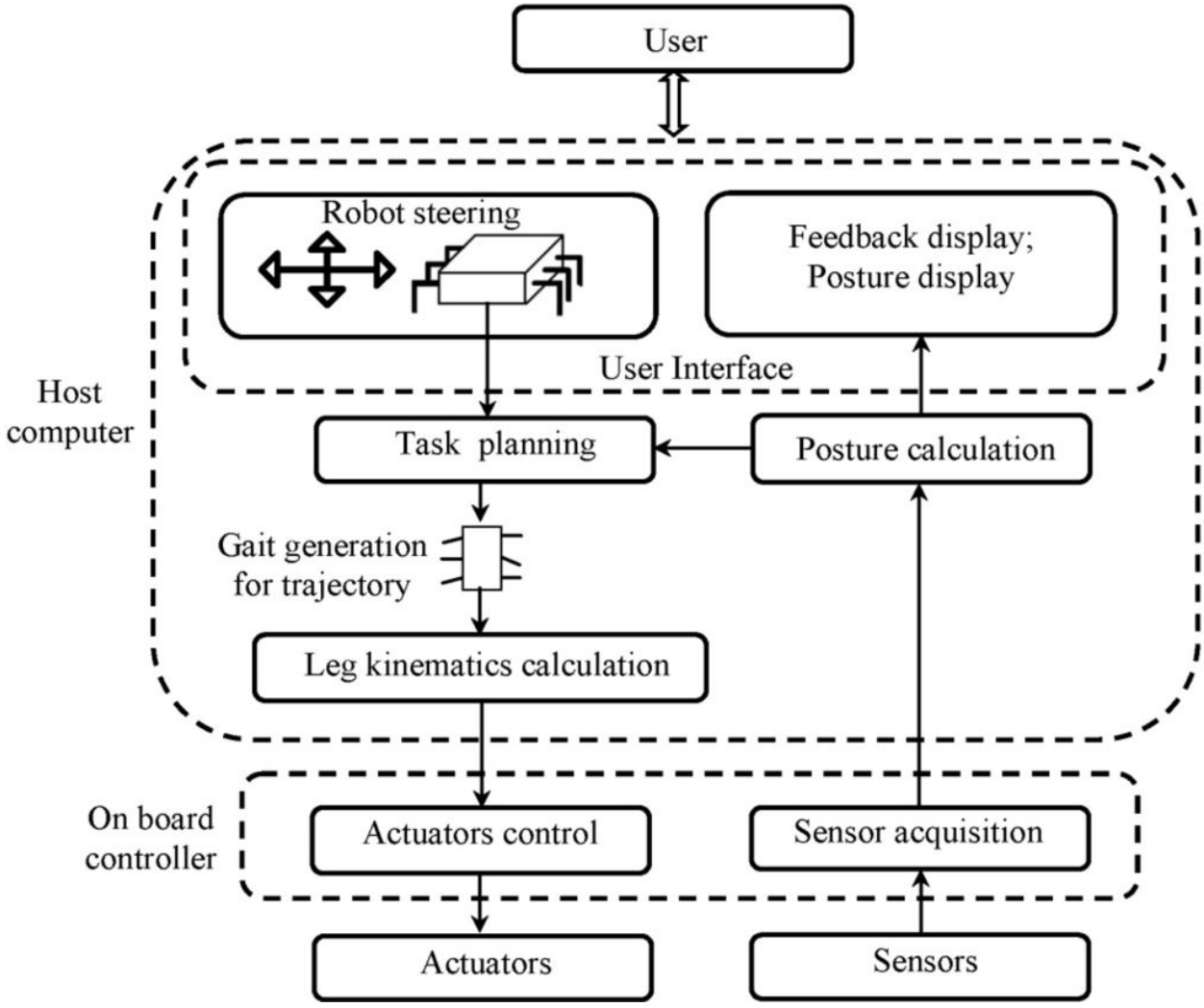
Storage decision #5

Issue	The composition of the hardware for storing sensitive health information.
Decision	The health information storage will be separated from the rest of the information the robot needs to store as it contains very personal information. It will have its own separate piece of hardware that is difficult to access with the proper procedure. The system will “revoke access” (The data on the drive will be deleted) to the storage unit in the case of an attempt at unauthorized access to the device.
Status	Pending
Group	Storage
Assumptions	-
Constraints	The component will be more difficult to service.
Positions	Hardware level drive encryption. Why not: This position was rejected because it did not provide adequate security compared to the chosen position. While this position would not have any of the downsides of the chosen solution it would not have provided satisfactory security. Hardware (and software) level checking for tampering with the system. Why not: although this position in itself is a good way of ensuring hardware integrity it alone is not enough to provide adequate security. Instead it was decided to implement this as part of the whole solution (see storage decision #5). Storing health information in the same hardware as the rest of the robot’s data. Why not: it was decided to store data on different storage devices because the security constraints are much more strict for the medical data and it would not permit the types of data to be mixed on a single storage device, therefore this position cannot be used. Also the other types of data stored on the robot will be accessed more often than

	the medical information and would benefit from increased performance (by not having to adhere to security constraints)
Argument	The medical data stored on the robot should not be accessible via means of a simple harddisk component. As this would open the robot up to attacks if the robot was ever disassembled or captured. Therefore it was chosen to encapsulate the storage device in an encasing hardware component to make sure the data (even though it's encrypted) cannot be accessed during maintenance. Instead the hardware component containing the storage device will be able to be swapped as a whole by the service engineer instead.
Implications	Due to the security concerns there has been more emphasis on the protection of stored information at a hardware level. While these hardware changes make the robot easier to service (due to the swappability of the secure storage component) but harder to test and maintain due to the black box nature of the secure storage component. Although a downside of checking for hardware tampering is that the system that checks for potential tampering should also be monitored against hardware tampering so a redundancy must be built in to safeguard against these types of attacks.
Relation Decisions	Storage decision #4
Related Requirements	The storage of medical information
Related Artifacts	Hardware and software storage mechanisms
Related Principles	Confidentiality, integrity and availability (CIA)
Notes	Architecture Tactics: Separation of Entities, Revoke Access, Detect Intrusion

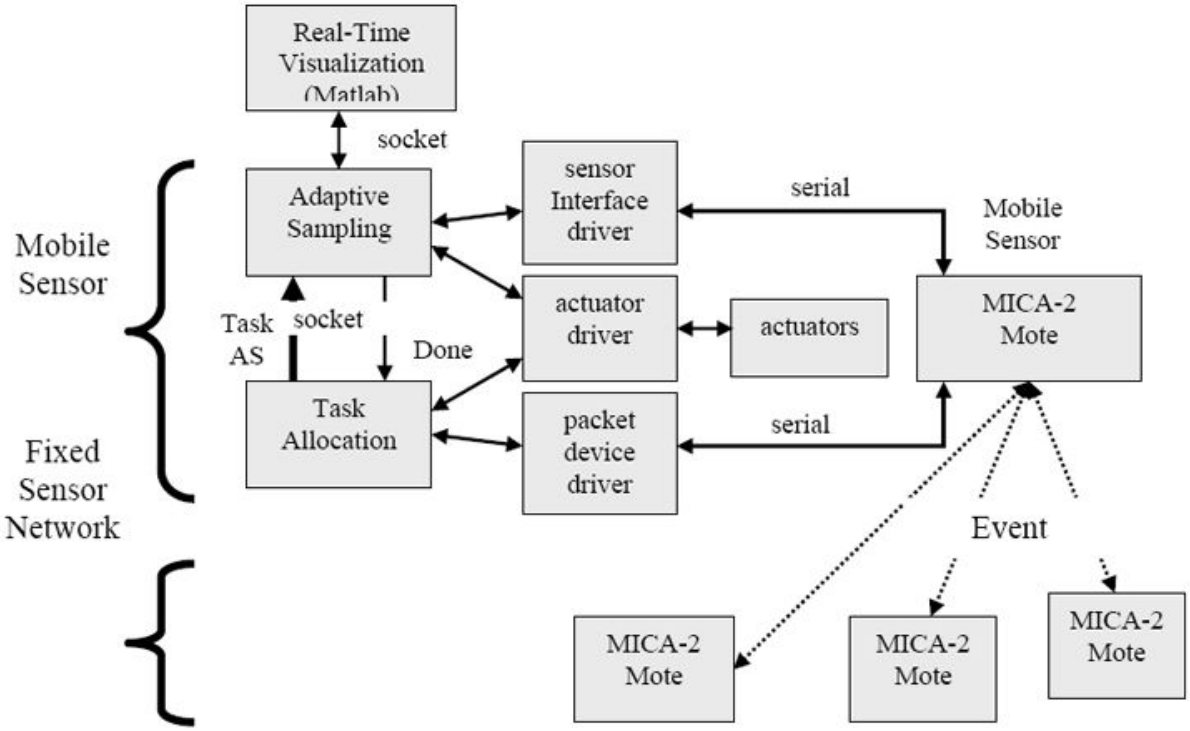
Appendix I: Reference Architectures

Reference architecture 1:



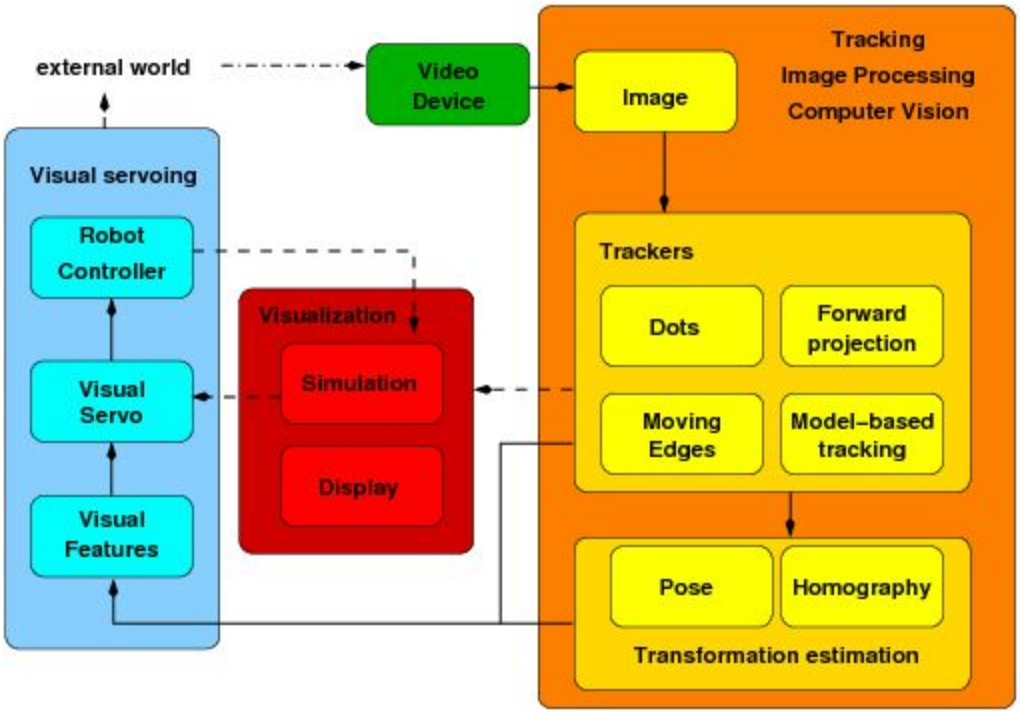
Source:http://www.mdpi.com/robotics/robotics-03-00181/article_deploy/html/images/robotics-03-00181-g007-1024.png

Reference architecture 2:



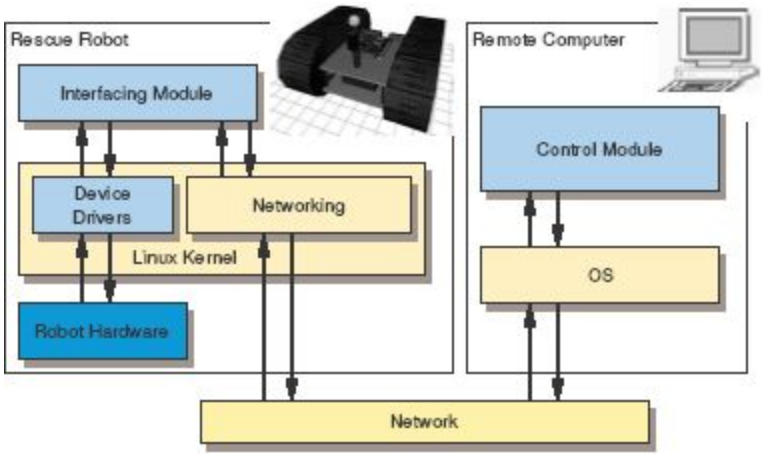
Source: http://research.cens.ucla.edu/projects/2005/NIMS/software_arch/Figure1.gif

Reference architecture 3:



Source: http://raweb.inria.fr/rapportsactivite/RA2009/lagadic/1.png

Reference architecture 4:



Source: http://www.eng.newcastle.edu.au/eecs/fyweb/Archives/2004/c2007000/images/softwareA.png

Reference architecture 5:

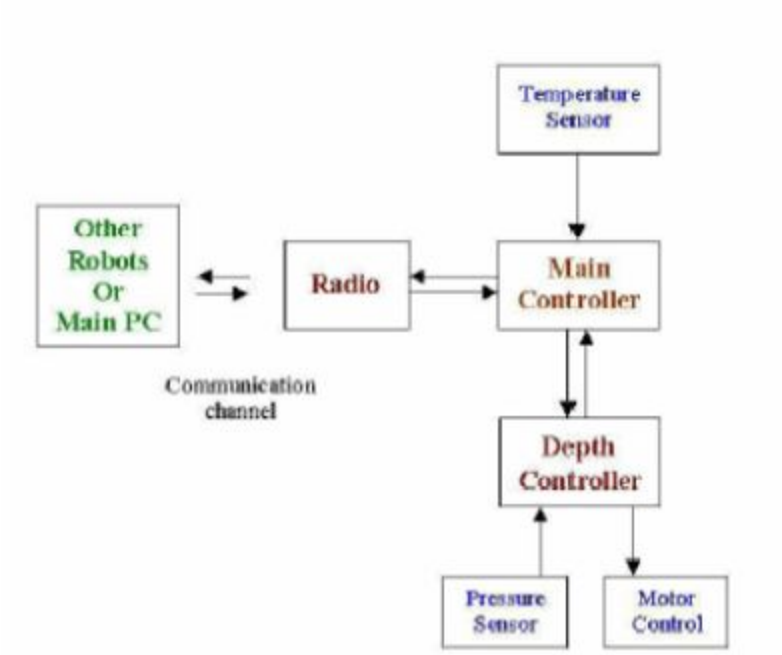


Figure 25: Software Architecture Schematic

Source: http://cens.ucla.edu/projects/2005/Actuation/testbeds/Figure25.GIF

Appendix II: Reflection implementation week 2

In week 4 our group was assigned to implement some examples using a pipe filter pattern. However the pipe filter implementation was not our own but one of our colleague's. We were supposed to use their implementation to create a number of example functions. We learned that working with someone else's codebase is quite a different experience than working on your own.

The implementation we were provided was reliant on the newest Java Development Kit (JDK) and the newest IDE versions. These technology choices were fine however none of our teammates had any experience with working with these new features in the JDK. Although this was no obstacle whatsoever it did give us some perspective on whether to use the newest technologies or staying with some older technologies of which knowledge may be assumed. It is important to specify the knowledge requirements in the documentation when creating components that will be reused by others. The last assignment made it clear that patterns shouldn't be forced to fit a design and they should be used when there is reason to.

To summarize:

- It is (sometimes) difficult work with the codebase of another team
- The choice of using established and new versions of technologies is complicated
- The knowledge required to use a component should also be documented
- Patterns should not be forced to fit into any particular design. They are only solutions to certain problems and always require a clear rationale before they should be applied to a design.

Link to implementation:
https://github.com/felixbarten/software-architecture/tree/master/ft_7_code/dev/src/nl.uva.se.funcpipes

Appendix III: Links and references

[1] Jeff Tyree, Art Akerman, "Architecture Decisions: Demystifying Architecture", IEEE Software, vol. 22, no. 2, pp. 19-27, March/April, 2005