Architecture of Distributed Systems 2017-2018

Homework assignment 1

R.H. Mak

Exercise

Consider the models on the following slides and answer the following questions. Each model is provided with a hyperlink to acknowledge its source and for additional information.

1. What building blocks do you see? What do they represent? Are they conceptual or physical?
2. Same questions as 1, but now for connectors?
3. To which views (1..*) does the model belong?
   Motivate why, and identify corresponding stakeholders and their concerns.
4. Which of the following EFRs are addressed (Y + motivation | N)?
   Performance/scalability, availability/reliability, security, maintainability, other?
5. Is there a concept of distribution (Y + motivation | N)?
6. Comment on the clarity/semantics of the diagram
   ☺ | 😐 | ☹ , plus motivation

Keep you answers crisp!

Building blocks:

Connectors:

View – concern – stakeholder (1..*):

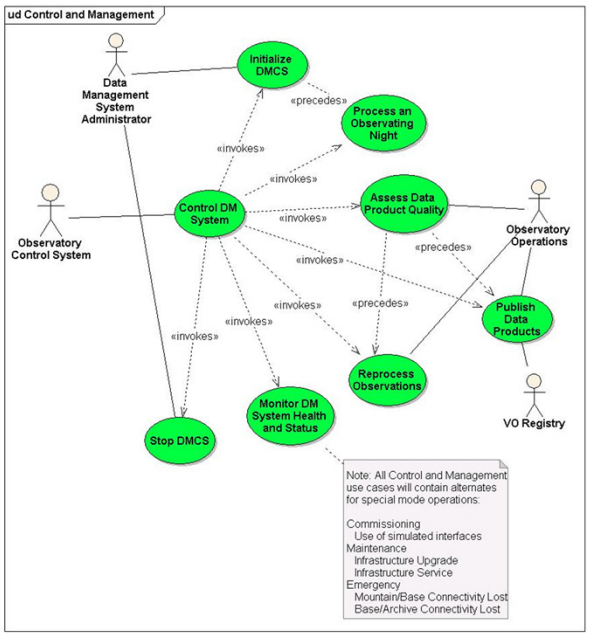
Extra-functional requirements (Y + motivation) /N :

- Security:
- Availability & reliability:
- Maintainability:
- Performance and scalability:

Distribution (Y + motivation) /N :

Clarity/Semantics (☺ | 😐 | ☹) + motivation:

Taken from:
https://dev.lsstcorp.org/trac/wiki/RobynsDraftFile1



5-Oct-17 2IMN10-HW1 3

Building blocks: (all C)

- *Actors both systems, OCS , OO and VO reg, and humans , DMSA (role)*
- *Use cases: about data management and control of a telescope*
- *Annotation with the nature health monitoring during various lifecycle stages*

Connectors: (all C)

- *Relationships between actors and use cases, indicating involvement*
- *Relationships between use cases, such as "invokes" or "precedes"*

View – concern – stakeholder (1..*):

- *Logical view describing the principal functionalities of the Control and Management of an Observatory,*
    - *of interest to its users, but also to acquirers, developers and communicators that have to explain the purpose of the system*

Extra-functional requirements (Y + motivation) /N :

- Security: *N*
- Availability & reliability: Y

    - *there is a use case that considers system health*
- Maintainability: *N*
- Performance and scalability: *N*

Distribution (Y + motivation) /N : *Y,*

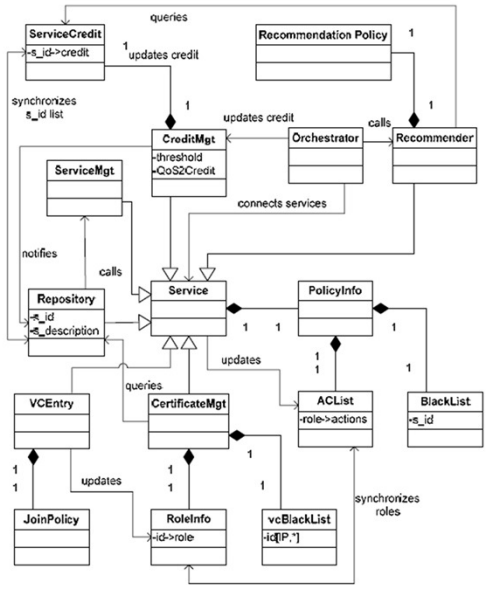
- *the whole system consists of a very big sensor (the telescope) and a lot of hardware and software to operate and store and process its data*

Clarity/Semantics (☺ | 😐 | ☹) + motivation: ☺

- *The diagram conforms to the UML conventions for use case diagrams*

Taken from:

VICSDA: using virtual communities to secure service discovery and access



5-Oct-17 2IMN10-HW1 4

Building blocks: *all (C)*

- *Classes that show the organization of an overlay network that can be imposed on top of a SOA and*

  that maintains virtual communities as a means to have secure discovery and access of its services.

  The overlay itself also has a SOA and those are described in this model.

  - *There are classes that represent Services and that contain rules (Policies) and data (Lists)*
  - *There are classes that perform framework tasks (..Mgt, Orchestrator)*

Connectors: *all (C)*

- *Is-a relationships: to indicate specialized services*
- *Part-of relationships: for a service the policies and lists containing rules and data needed to provide its service*
- *Other relationships (associations) that indicate interaction (calls, queries, notifies, updates, synchronizes)*

View – concern – stakeholder (1..*):

- *Logical view that addresses all stakeholders who need the have a global overview of the system, such as end-user and developers, but also system integrators and testers which may want to use this overlay in combination with an existing SOA.*
- *Development view: interesting for programmers to see how functionality is*

partitioned; would have been more interesting for them if more information on interfaces would be present.

Extra-functional requirements (Y + motivation) /N :

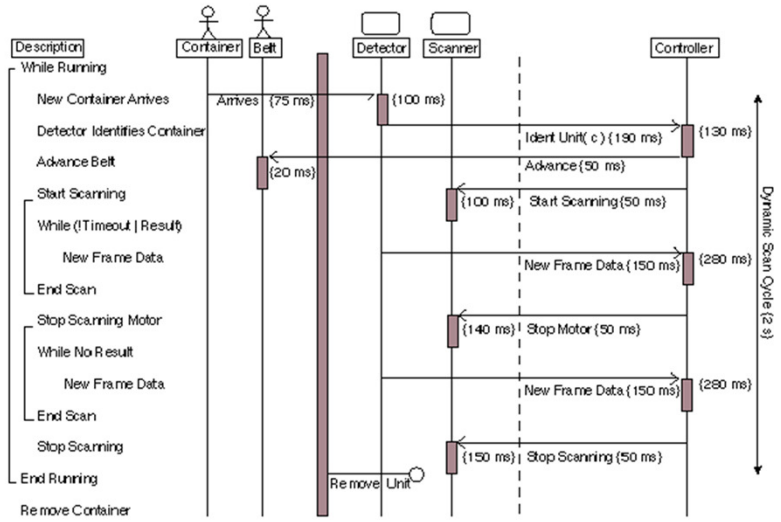
- Security: *Y*
    - *There are BBs for access control (ACList, BlackList)*
- Availability & reliability: *N*
- Maintainability: *Y*
    - *It shows how the functionality of the VC-overlay is distributed over classes which helps locating code that needs to be modified upon change requests*
- Performance and scalability: *N*

Distribution (Y + motivation) /N : *N*

Clarity/Semantics (☺ | 😐 | ☹) + motivation: 😐

- *This is a classical UML class diagram, for which the semantics is clear.*
- *To understand the names of the classes and their relationships, however, requires reading at least the abstract of the paper from which the diagram is taken.*
- *Possibility to add services only clear from paper*

Taken from:
http://werner.yellowcouch.org/Papers/rtuml/#toc9



5-Oct-17 Rudolf Mak TU/e Computer Science 2IMN10-HW1 5

Building blocks:

- *Container and belt (P)*
- *Detector, Scanner (C)*
- *Lifelines and activities such as container identification, scanning, control (C)*
- *Legend in the form of a textual process description (C)*

Connectors:

- *Asynchronous messages triggering activities (C)*

View – concern – stakeholder (1..*):

- *Process view, of interest to system integrators and testers that e.g. can verify whether timing constraints are met.*

Also of interest to programmers that must ensure that their code for activities meets the timing constraints.

- *It is also a scenario describing the handling of a single container on the belt interesting all stakeholders that want to have a more detailed understanding about the process. For users it defines functionality of the system*

Extra-functional requirements (Y + motivation) /N :

- Security*: N*
- Availability & reliability: *N*
- Maintainability: *N*
- Performance and scalability: *Y,*
    - *The diagram contains timing information w.r.t. the duration of activities and latency of messages*

Distribution (Y + motivation) /N : *Y,*

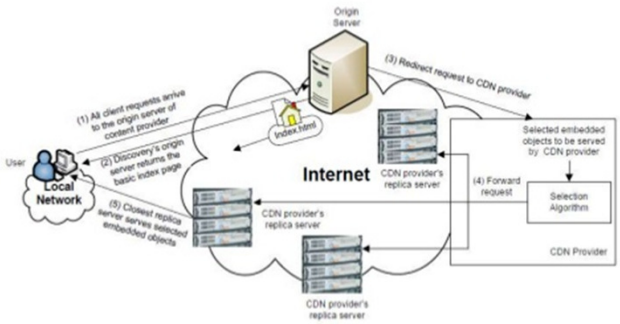
- *We see various components, performing distinct tasks.*

Clarity/Semantics (☺ | 😐 | ☹) + motivation: 😐,

- *Clearly drawn sequence diagram, however, with a few peculiarities.*
    - *Description which is sort of a legend and helps in understanding.*
    - *On the other hand, there is a mysterious vertical bar separating the physical object from the conceptual ones*
    - *No interaction between scanner and detector???*

Taken from:
http://https://www.slideshare.net/shivrock87/content-delivery-network-17487498

Request Routing in CDN Environment



5-Oct-17 Rudolf Mak TU/e Computer Science 2IMN10-HW1 6

Building blocks:

- *User situated in a local network (P)*
- *Origin server (of the content provider) (P)*
- *The CDN provider (C)*
- *CDN provider's Selection algorithm (C)*
- *Replica servers of the CDN provider (P)*

Connectors:

- *Messages, requests for content and replies (C)*
- *Data stream of embedded objects (C)*

View – concern – stakeholder (1..*):

- *Process view, because it explains the sequence of request and replies needed to deliver content at a user.*
    - *of interest to system integrators, but also to the customer , acquirer of the system which is probably the content provider and the content consumers (users) to see how quality of service is achieved and the supplier of the CDN*

- *Physical view, because it shows physical components*
    - *Of interest to system engineers although it contains very little information*
- *In general of interest to communicators that need to explain the organization and the process of content delivery*

Extra-functional requirements (Y + motivation) /N :

- Security: *N*
- Availability & reliability: *Y,*

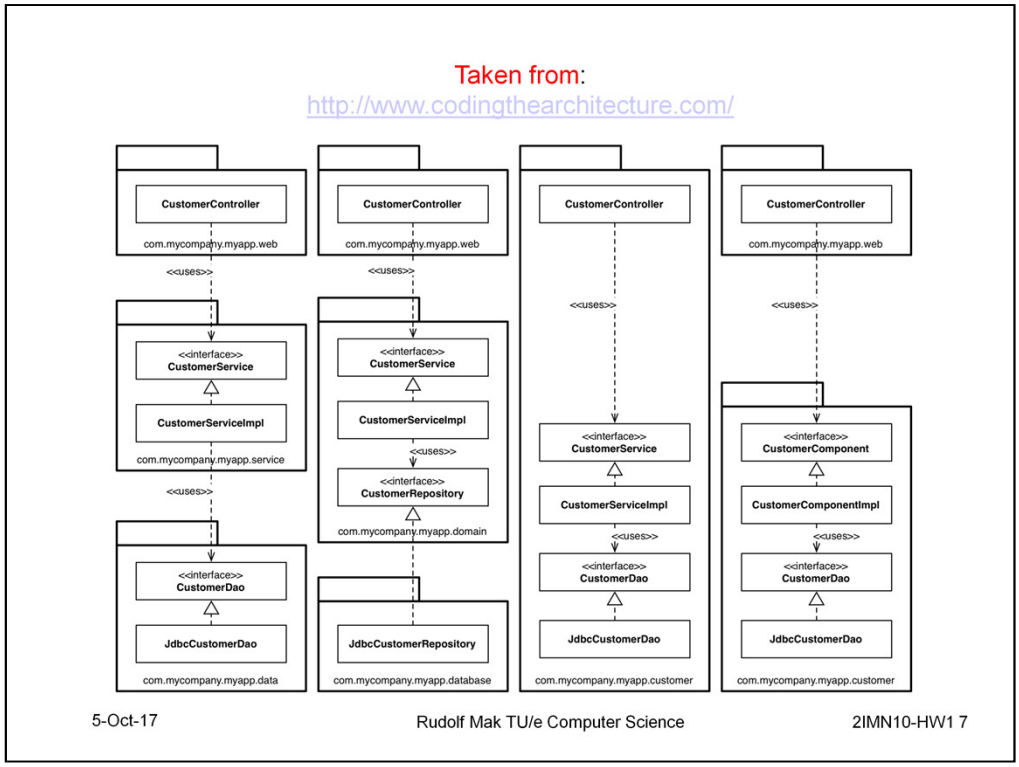
  multiple replicas of content increase its availability to users.
- Maintainability: *N*
- Performance and scalability: *Y,*
    - *users obtain content from a nearby replica server to reduce communication latency and saving network bandwidth.*

Distribution (Y + motivation) /N :Y

- *Content of origin server is replicated (cached) at geographically distributed replica servers requiring the various communicating entities to be connected through the internet.*

Clarity/Semantics (☺ | 😐 | ☹) + motivation: ☺

- *This is a collaboration diagram, using clear icons to explain the nature of entities and sequence numbers to indicate the routing order.*



Building blocks: (all C)

- *Layers, each vertical layer is a package identified by URL. Layering is strict!*
    - *Horizontal grouping is an enumeration of layering options!*
- *Modules containing Interfaces and/or Implementations*
    - *B.t.w. Dao stands for data access object*

Connectors: (all C)

- *Containment of modules into packages/layers*
- *Dependencies (usage) between modules in the same or in distinct packages*
- *Implementation relationships*

View – concern – stakeholder (1..*):

- *Development view, of interest to programmers / software developers indicating how they could organize their code. Also of interest to users to see what is available in each option.*
- *The diagram is also of use to communicators /educators.*
    - *In fact the diagram is used to explain to programmers the consequences of the different layering options.*

Extra-functional requirements (Y + motivation) /N :

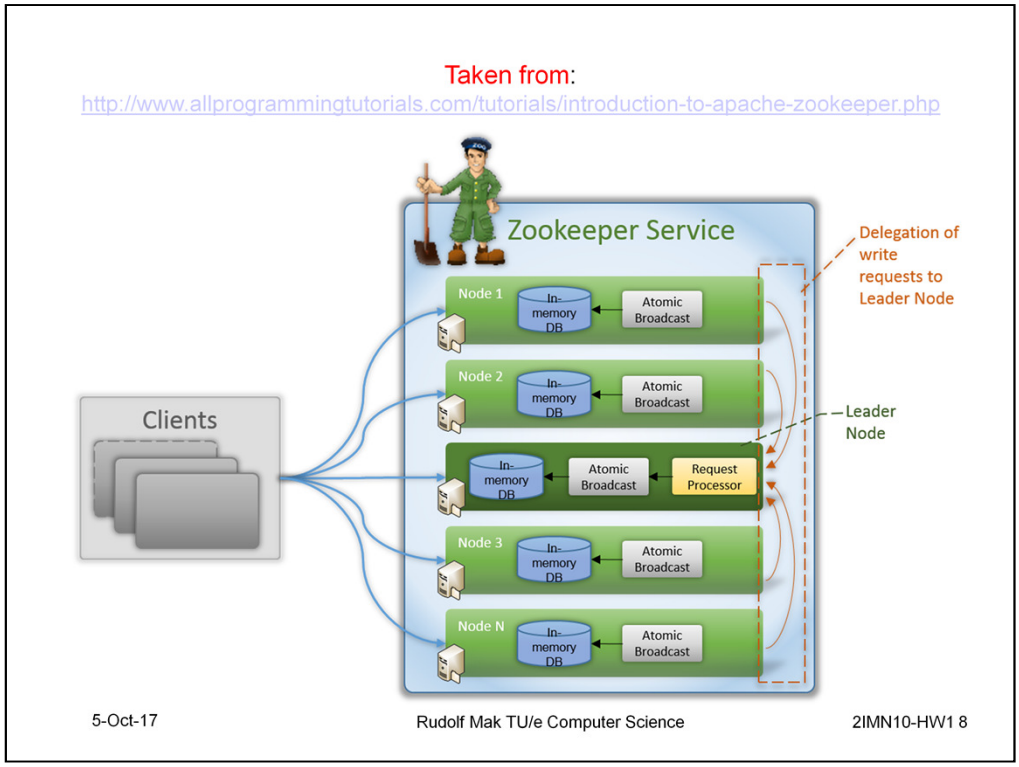
- Security: *N*
- Availability & reliability: *N*
- Maintainability: *Y,*
    - *proper organization of the code with clearly indicated (and relatively few) dependencies helps in developing and maintaining the software.*
- Performance and scalability: *N*

Distribution (Y + motivation) /N : *N,*

- *This is about the code organization. Names do not reveal distribution aspects of the system.*

Clarity/Semantics (☺ | 😐 | ☹) + motivation: ☺

- *A very clear package diagram, showing 4 distinct ways of layering application software.*
- Use of different names (Repository vs Dao & Service vs Component) in different options is confusing.



Building blocks:

- *Client nodes and Zookeeper nodes (P)*
- *Zookeeper service (C)*
- *In memory storage (P)*
- *Atomic broadcast process (C)*
- *Request processor (P | C)*
    - *A processor is a physical entity, but here it seems more likely that a process handling requests is meant*

Connectors: (all C)

- *Client requests*
- *Delegations of write requests*
- *Read/write operations on the DB invoked by the atomic broadcast processes.*
- *Invocation of atomic broadcast by request processor.*

View – concern – stakeholder (1..*):

- Logical view, of interest to programmers and system integrators who need to realize the proper handling client requests. Users can see replication is offered

and that each interaction is with a single node.

- Physical view, of interest to system engineers who need to provide a fault tolerant hardware configuration, although the diagram hardly contains helpful information for that.

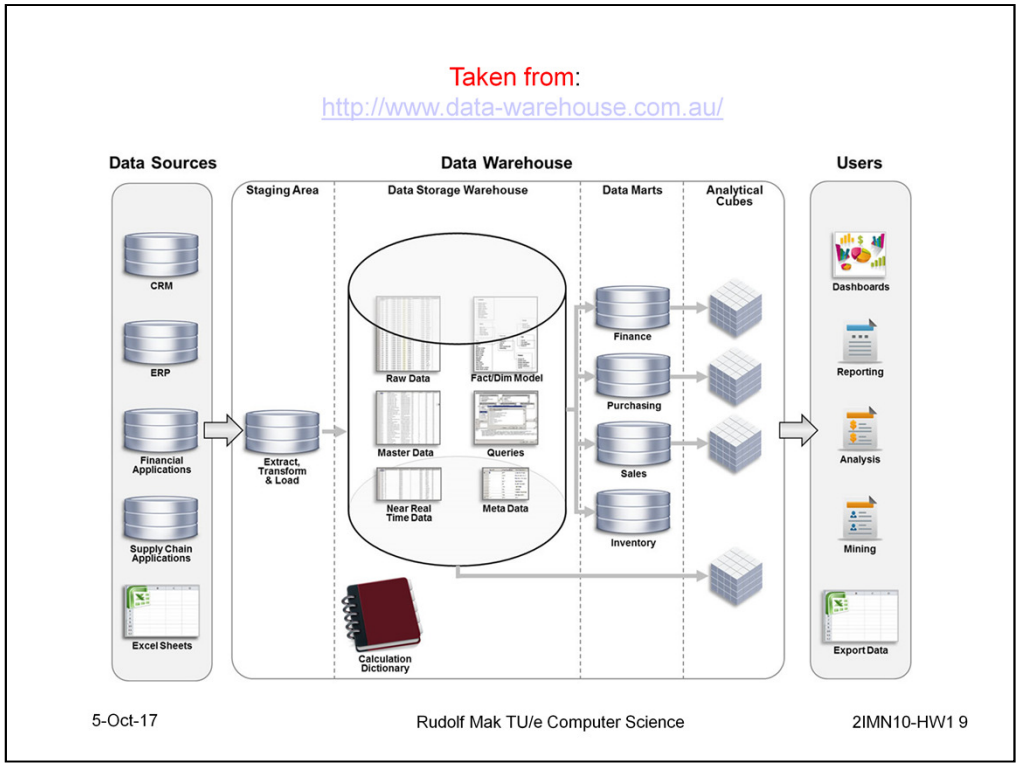
Extra-functional requirements (Y + motivation) |N :

- Security: *N*
- Availability & reliability: *Y,*
    - *the Zookeeper nodes are replicated to provide improved availability and reliability through redundancy*
- Maintainability: *N*
- Performance and scalability: *Y,*
    - *Each node will answer a read request (can implicitly be derived from the diagram, because only write requests are delegated). This reduces response time for read operations.*
    - *In memory storage of a database for performance enhancement*

Distribution (Y + motivation) /N : Y, *Zookeeper has N physically distinct nodes*

Clarity/Semantics (☺ | 😐 | ☹) + motivation: ☹,

- *No obvious drawing conventions. And as a result poor semantics.*
- *Pre-knowledege of the zookeeper service is needed to understand this diagram.*
- *Flow of control of read operations by the clients is not properly indicated.*
- *Communication by atomic broadcast is left implicit.*



Building blocks:

- *Software systems, such as CRM, ERP, (C) that are data sources, organized in the Data Sources layer (C)*
- *Datawarehouse Layer (C) consisting of an ETL system (C) , a large data store (P), data marts (C), i.e. specialized data analysis processes producing*

analytical cubes (C), preprocessed data similar to pivot tables that can be used by other applications.

- *User applications (C) organized in a Users layer (C)*
- *Calculational dictionary (C), no clue what this represent!*

Connectors:

- *Arrows indicating workflow (C), i.e., production /transformation/analysis/consumption of data.*
- *Hence the arrow also indicate dataflow between stages (pipelining)*

View – concern – stakeholder (1..*):

- *Context view, because it shows the context of a data-warehouse:*

  the systems that generate data and the user applications that make use of the processed data, analytical cubes of the

- *Process view, because it shows the flow of data and the way its processed*
- *Logical view, because it explains the general organization of a data-warehouse.*

In general of interest to all stakeholders (acquirers, data analysts, ...) that need to understand the organization of and the functionality offered by a data-warehouse

Extra-functional requirements (Y + motivation) /N :

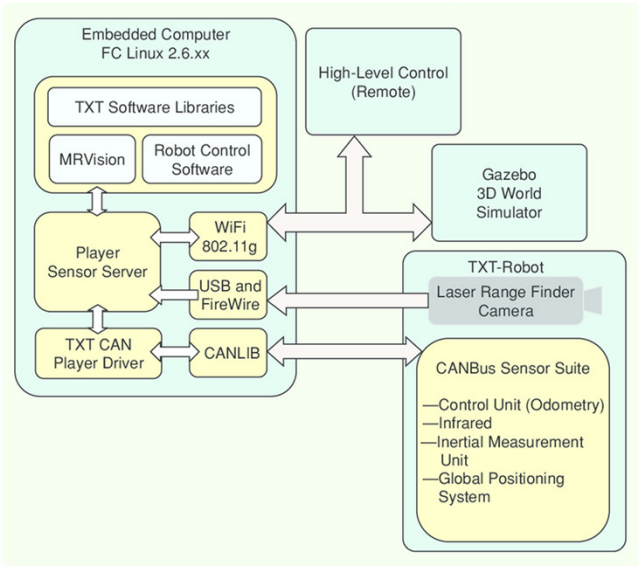
- Security: *N*
- Availability & reliability: *N*
- Maintainability: *Y*
- Performance and scalability: *Y*

Distribution (Y + motivation) /N : *Y*

Clarity/Semantics (☺ | 😐 | ☹) + motivation: 😐,

- *Semantics of calculation dictionary is unclear,*
- *Also the icons for processes and data are very clear.*

Taken from:
https://www.researchgate.net/figure/3207861_fig2_FIGURE-6-Vehicle-software-architecture-This-block-diagram-shows-how-the-onboard-sensor



Building blocks:

- *Devices: embedded computer, remote control, simulator, TXT-robot (P)*
- *Robot sensors, such as LRF-camera (P) and may be some others (Odo, IMU, GPS) whose presence can only be guessed from control software*
- *Software modules: TXT libraries, vision and control, communication modules, driver, Sensor suite (C)*

B.t.w. the robot is a vehicle! See added picture.

Connectors:

- *Communication between devices and interactions between software modules on the same device (C)*
- *Part of relation: LRF camera (sensors in general) is part of the robot (C)*

View – concern – stakeholder (1..*):

- *Deployment view, of interest to system engineers who are concerned with communication technologies between the devices and configuring the robots.*

In general to users and other stakeholders who want to understand the structure of the system.

- *Physical view (technologies, CAN-bus ,TXT-Robot, LRF camera, Gazebo*

Simulator) of interest to suppliers.

Extra-functional requirements (Y + motivation) /N :

- Security: *N*
- Availability & reliability: *N*
- Maintainability: *N*
- Performance and scalability: *N*

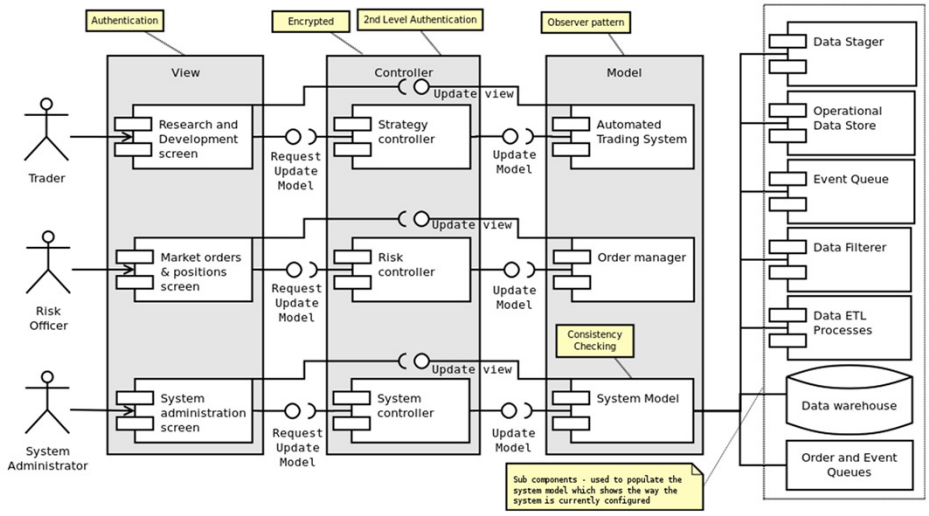
Distribution (Y + motivation) /N : *Y*

- *The system has multiple physical components (Control is remote!, but the computer is embedded on the robot)*
- *Although we only see the description of a single robot and its environment, many applications involve multiple robots*

Clarity/Semantics (☺ | 😐 | ☹) + motivation: 😐,

- *Reasonable w.r.t. to deployment, but role of simulator is obscure. One needs to read the paper to understand this.*

Taken from:
http://www.turingfinance.com/algorithmic-trading-system-architecture-post//



5-Oct-17 Rudolf Mak TU/e Computer Science 2IMN10-HW1 11

Building blocks: (all C)

- *Human actors*
- *MVC-Layers*
- *Components: screens, controllers, models*
- *Subcomponents of the System Model: Data Stager, ..., O&E queues*
- *Annotations*

Connectors: (all C)

- *Interface bindings (lollipop-recepticle pairs)*
- *User interaction with screens of the view layer*
- *Inclusion (part-of) relations (subcomponent of)*

View – concern – stakeholder (1..*):

- *Logical view, of interest to the users of the system concerned with the functionality offered by the system*
- *Development view, of interest to programmers, system developers icncerned with the organization of the software*

Extra-functional requirements (Y + motivation) /N :

- Security: *Y,*
    - *encryption and authentication indicated*
- Availability & reliability: *N*
- Maintainability: *Y,*
    - *Layering, MVC pattern, strangely each model supports only one view!*
- Performance and scalability: *N*

Distribution (Y + motivation) /N : *Y,*

- *In the MVC pattern the views are usually situated on other machines than the models*

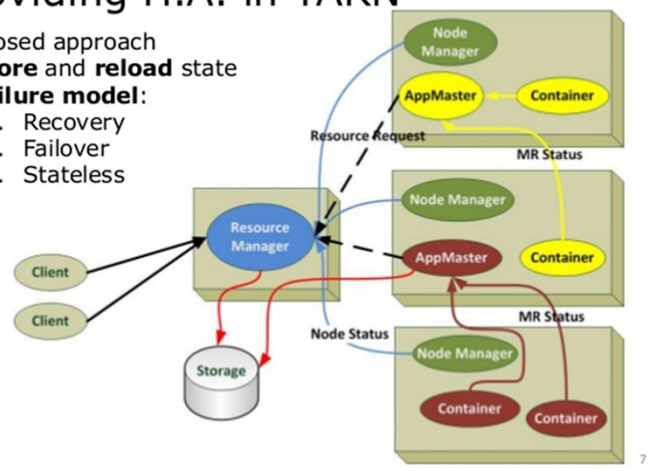
Clarity/Semantics (☺ | 😐 | ☹) + motivation: ☺,

- *UML component diagram, clear y describing the MVC-pattern.*

Taken from:
https://www.slideshare.net/arinto/high-availability-in-yarn

Providing H.A. in YARN

Proposed approach
- **store** and **reload** state
- **failure model**:
  1. Recovery
  2. Failover
  3. Stateless



5-Oct-17 Rudolf Mak TU/e Computer Science 2IMN10-HW1 12

Building blocks:

- *Nodes: clients and master and worker nodes (P)*
- *Data store (P)*
- *Components such Managers, Appmasters and containers (C)*
- *Legend for state and failure models (C)*

Connectors:

- *Client requests (C)*
- *Status updates (C)*
- *Resource requests (C)*
- *Reads and writes of state data to permanent storage (C)*

View – concern – stakeholder (1..*):

- *Deployment view, of interest to system engineers, administrators who are concerned where YARN runtime platform components (the managers) need to be deployed.*

Also of interest to cluster users (clients running applications) to see how their

applications are distributed and how resources are allocated

Extra-functional requirements (Y + motivation) /N :

- Security: *N*
- Availability & reliability: *Y*,
    - *Failure models are named and state of resource manager and app masters is stored for crash recovery.*
- Maintainability: *N*
- Performance and scalability: *Y*,
    - *We see parallellism/concurrency in the form of applications running on multiple nodes*

Distribution (Y + motivation) /N :*Y,*

- *YARN is a cluster manager that involves components on many physical nodes.*

Clarity/Semantics (☺ | 😐 | ☹) + motivation: 😐,

- *Unclear what is meant by MR-status*
- *Meaning of the colors unclear*
- *Difficult to link the legend to the diagram*