Eindhoven University of Technology

Computer Science and Engineering

Architecture of Distributed Systems

# Homework Assignment 1

*Authors:*
Jasper Selman
Ramon de Vaan

September 8, 2015
Eindhoven

## 1 Model 1 - Saas-Mode

1. The building blocks we see are the firewalls (conceptual). The purpose of these firewalls are obviously security related, such that the site is protected from malicious attacks. We also see load balancers and filtering. These are for work distribution and are also conceptual. The other components are all physical, namely the servers (application and database), databases and NAS devices. These are for storing and sharing the information on the site.

2. You see the red lines connecting components within a site to each other. This represent how components in a site are linked to each other and with which components they can communicate. You also see the green arrows, these connectors show how the main site communicates with the secondary site(s). All these connectors are conceptual.

3. We see a clear Design view with regards to the communication. You can clearly see which components should interact with other components. You can also clearly see the Deployment view. You can clearly see all the components in the view and it gives an overview of the services.

4. **Performance/Scalability** Yes, the use of the filtering and clustering give notion of the performance of the model. Also the scalable parts are shown in groups from 1, 2 to N.

   **Availability/Reliability** No.

   **Security** Yes, the use of firewalls and synchronous replications define the security.

   **Maintainability** No.

5. Yes, You can clearly see the flow throughout the system. You see load balancers, applications, filters, Clustered Databases, Severs and NAS devices and how these are linked to each other.

6. You can clearly see what all the components in the sites are and how they are connected. Different components have there own colouring and smaller parts of the model, belonging to each other are structured in a group (dashed lines). Different ways of communication are shown with different arrows. Not really clear what the model does precisely and what users/systems are using it.

## 2 Model 2 - VICSDA

## 3 Model 3 - AUTOSAR

1. The building blocks we see in this model are all the components in the RTE system. They are divided in several categories, namely pink blocks for drivers, green for abstractions, blue for services. We also see some grey blocks. These are external OS-applications or Micro-controllers. At last we also have blue blocks. These are "actors" in the error throw part. So they are Receivers of the errors or Senders. All these building blocks are conceptual.

2. The connectors we see are the arrows from certain parts to other parts indicating the errors which are thrown by the hardware and/or software. They are all conceptual.

3. We see a bit of an implementation view. All the parts which have to be implemented in the RTE are shown, but how they are implemented and what they do however is left free to the developer. We also see a bit of the Process view, mostly how the errors have to be handled and from which part they came.

4. **Performance/Scalability** Yes, they try to address the performance. Libraries are used and every part of the software, like communication, memory, I/O and on-board devices have their own abstraction, drivers and services. I do not know however how well this model scales when you enlarge it.

   **Availability/Reliability** Yes, a lot of errors are handled and pointed out.

   **Security** No.

   **Maintainability** No.

5. A little bit. We know that parts of the software, like memory, I/O, communication, etc. have their own services, abstractions and drivers, but we do not really know how this is distributed in the RTE so we do not know a lot.

6. I think this is not a very clear diagram, because it looks kind of messy at the start. This is because there are a lot of blocks and lines all over these blocks and on the right is some extra text. But when you look a little longer at the diagram you see that actually it is not that bad at all. All the blocks which have something in common, like OS, drivers, services etc. are coloured in the same way. Errors are always shown as a red lightning flash and the text on the right is the legend for the errors. Besides that I am missing the context of the model. I know that it has something to do with an automotive system from the URL and that a lot of errors are handled (or at least mentioned) but I do not really know what the system does.

## 4 Model 4 - Collaboration Diagram

## 5 Model 5 - LinkedIn

1. The blocks in this model are the applications, the services, the "cloud", and all the databases and servers. The databases and servers are all physical and of course for storage of all the LinkedIn data. The applications, services and cloud are all conceptual. The applications represent the profiles of different users. The services represent what is used on LinkedIn, reading the news, communicate with other users, sharing something in a group, etc. The cloud is where all these web apps are used.

2. There is only one sort connector in this case, namely an arrow. Sometimes the arrow literally states what happens between the components (for example profile updates). Other times it is less obvious and you only know to components are connected and communicate. All the connectors are conceptual.

3. We see a clear Design view with regards to the communication. You can clearly see which components should interact with other components and if it should be a read/write

action or something else. You can also clearly see the Deployment view. You can clearly see all the components in the view and it gives an overview of the services. You can also see implementation things in the building blocks and connectors, so you also see the implementation view.

4. **Performance/Scalability** Yes, you see a stash of databases, and that every service has its own database. It is shown that this is extendible by the ... etc.

   **Availability/Reliability** No.

   **Security** No.

   **Maintainability** Yes, since every component has its own database, you can easily change some of the services or add/remove one.

5. Yes, you can clearly see the distribution between services and databases and which application uses which service.

6. You can clearly see all the components in the sites and how they are connected. Different components have there own colouring and smaller parts of the model. Communication is clearly shown with arrows, sometimes the arrows also note what the communication is. Not really clear where the architecture starts (I guess in the web apps). But over all very structured and easy readable and understandable.

## 6 Model 6 - NHIN

## 7 Model 7 - Vending Machine

1. The states, sub-states and superstates. They are all conceptual and model in which state the vending machine is after an event has occurred.

2. The arrows represented by the events. These are all conceptual. They model an event which would happen in real life.

3. **Process View** System integrators, testers can see how the various parts interact. Developers see what the machine has to do when something happens. They also see what they have to implement to see how the state changes.

   **Development View** The state-chart is used to create the model by developers.

4. **Performance/Scalability** No.

   **Availability/Reliability** Yes, there are error methods when an error occurs.

   **Security** No.

   **Maintainability** Yes, the chart is very clear and easy to maintain. It is both useful for implementation and modification.

5. No.

6. It is a very clear diagram. In development view it is clear in which states the machine can be and how these states can be reached. It is however free for the programmer how this is implemented. It is also very clear whether something is a state or an action. These are the only two possibilities in the chart. Furthermore are the sub-states very useful to see which components communicate with each other.

## 8 Model 8 - Consumer Website