Architecture of Distributed Systems

Homework Assignment 1
2019-2020

R. H. Mak

Technische Universiteit
Eindhoven
University of Technology

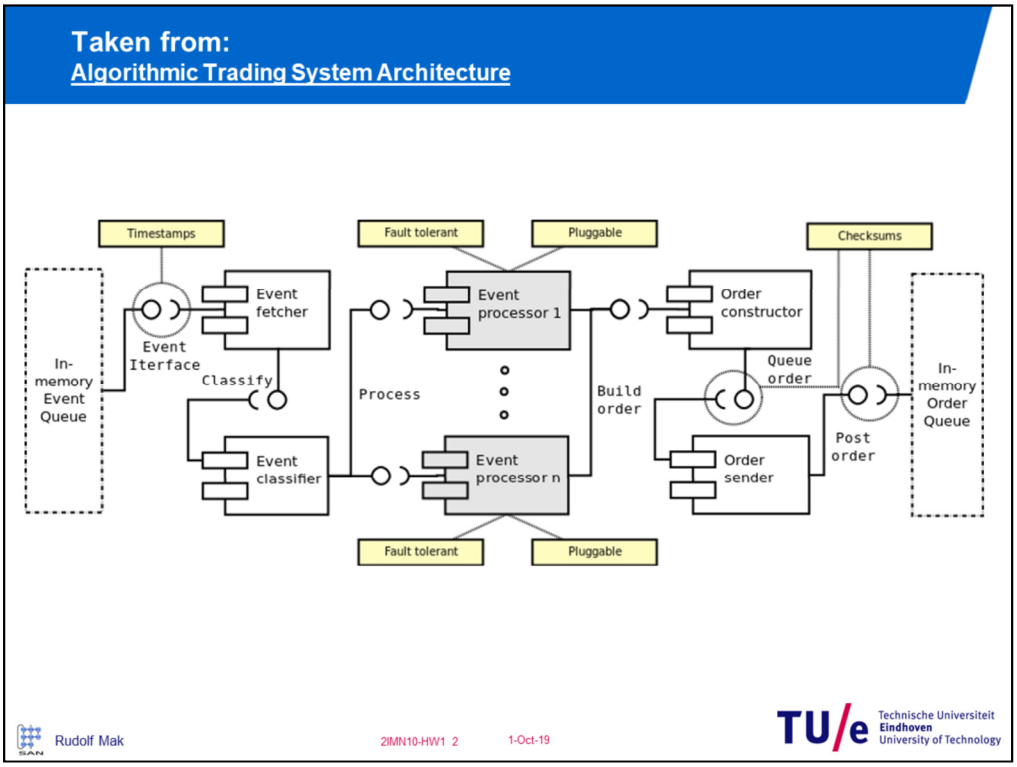
Where innovation starts

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/generic (C) or physical/specific (P)?
2. Same questions as 1, but now for connectors? Do not forget (C) or (P)!
3. To which viewpoints (1 up to 3) does the model belong?
   Motivate why, and identify corresponding stakeholders and their concerns.
   You may choose from both Kruchten and Rozanski-Woods libraries.
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 Which, what + (C | P): All C

- Components such as Event fetcher/classifier/processors and Order constructors/senders
- (temporary) Storage elements: event and order queues
- Annotations(comments) detailing aspects of elements

Connectors Which, what + (C | P): All C

- (Annotated and named) Interfaces, both provided (lollipop symbol) and required (receptacle symbol)

View – concern – stakeholder (1..*):

1. Process view, because we see the interaction between components, of interest to system integrators and testers who have to compose the system from its components
2. Development view. The identified components are also interesting for programmers especially to see where to implement future extensions (see EFR maintainability). Moreover the diagram (weakly) suggests that the components are organized in 3 functional layers (fetching and classifying, processing, and ordering)

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

- Security: N

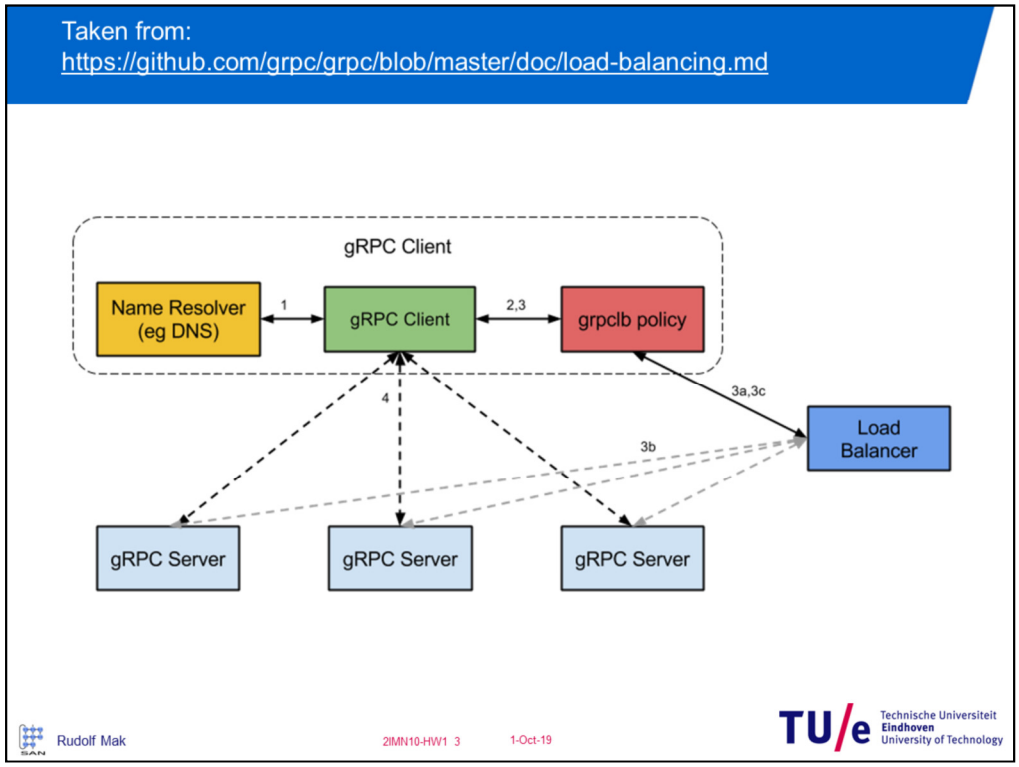
- Availability & reliability: Y,
    - Because the event processors need to be fault tolerant; moreover orders are equipped with a checksum which makes it possible to see whether the are corrupted
    - It is not clear whether the same event is processed by multiple event processors thus providing reliability through redundancy or whether each event processor merely deals with a different class of events.
- Maintainability: Y,
    - Pluggable event processors make it possible accommodate new classes of events in the future without modification of the overall system architecture
- Performance and scalability: Y,
    - Multiple event processors for a single event class will also improve performance through concurrency

Distribution (Y + motivation) | N: Y,

- Because there are multiple event processors

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

- The diagram uses classical UML notation for components and interfaces which improves readability. Semantics are not always as clear, see remark under reliability and also the nature of the events and orders is not indicated



Building blocks Which, what + (C | P):

- Servers (C | P) could be either a process or a dedicated machine running that process
- Processes/plug-ins (C): Name resolver, gRPC client (green), grpclb policy, load balancer
- There is also the dashed box (C), called gRPC client whose nature is unclear. I see 2 options:
    - It is a component and the yellow, green and red boxes are in fact plug-ins,
    - It is a host machine and the yellow, green and red entities are processes running on this machine

Connectors Which, what + (C | P):

- Remote procedure call (C) (request-reply) between a client and a server
- Messages (C) between the servers and a load balancer (indicating load, and availability?)
- Request-reply interaction (C) between the dark colored building blocks
    - All labelled with a number to indicate control flow

View – concern – stakeholder (1..*):

1. Process view, order of action in an RPC, of interest to system integrators and testers. Programmers may be interested in the opportunity to provide specific load balancing policies via a separate component.
2. Deployment view, the yellow green and red entities are deployed on the dashed gRPC client, Interesting to system administrators and operators who have are

responsible for configuration.

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

- Security: N
- Availability & reliability: Y,
    - Multiple gRPC servers not only improve performance by allowing concurrent access, but also availability. As long as not all servers fail, the RPC-service is available
- Maintainability: Y?,
    - Because plugins indicate to programmers where to put certain functionality. In particular, instantiation of policies.
- Performance and scalability: Y,
    - Because we see a load balancer, so the load of RPC calls is distributed over multiple servers.

Distribution (Y + motivation) | N: Y,

- because we see a client server architecture in which there are not only multiple client (which is not shown), but also multiple servers

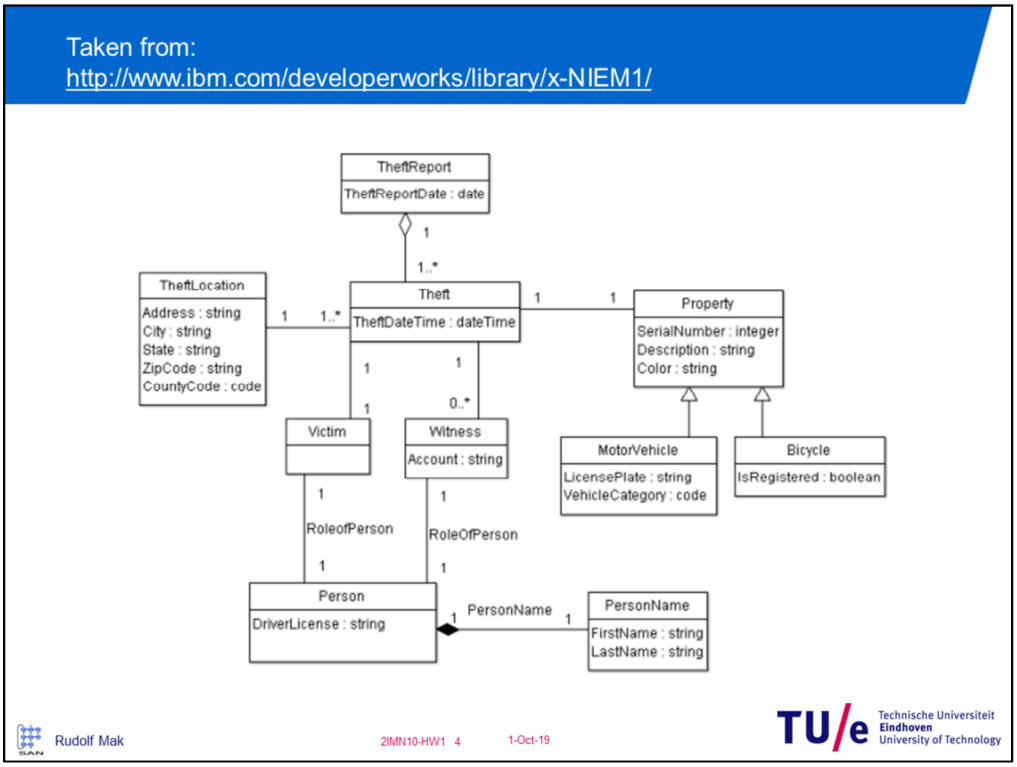
Clarity/Semantics (☺ | 😐 | ☹) + motivation: ☹

Similar to a UML-collaboration diagram, but different notation

Semantics especially w.r.t. load balancing unclear

Nature of communicated data is necessary to make sense of this diagram

gRPC client is used twice (see building blocks)



Building blocks Which, what + (C | P): (all C)

- Classes denoting the important concepts of a theft report. Attributes of these classes indicating the most important data recorded by the objects of the class

Connectors Which, what + (C | P): (all C)

- Aggregation (has a) relationships,
- Specialization (is-a) relationships,
- Composition (part-of) relationship (arrow with black diamond)
- Association relationship, e.g., capturing the role of a person in a theft

View – concern – stakeholder (1..*):

- Logical view: domain model showing the concepts involved in a theft report
    - End-users can find the information that should be provided to make a theft report
    - Acquirers of the system (police?, insurance companies?) can see whether this conforms to their common practice to make such reports
    - Developers can organize their implementations around these concepts
- Development view: Classes carry attributes that contain vital data that should be represented, which is of interest to programmers
- In the RW viewpoint library this model could also be classified as belonging to the context and information views

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

- Security: N
    - Access control and authorization of readers of these reports could be provided in a separate model.
- Availability & reliability: N
- Maintainability: Y
    - Software constructed to reflect the domain entities will help maintaining, modifying, and updating the software.
    - It shows where data is kept, thus suggesting where the responsibility of reporting that data could be located.
- Performance and scalability: N
- Other: the diagram helps understanding the domain, hence is of interest to all stakeholders for mutual communication.

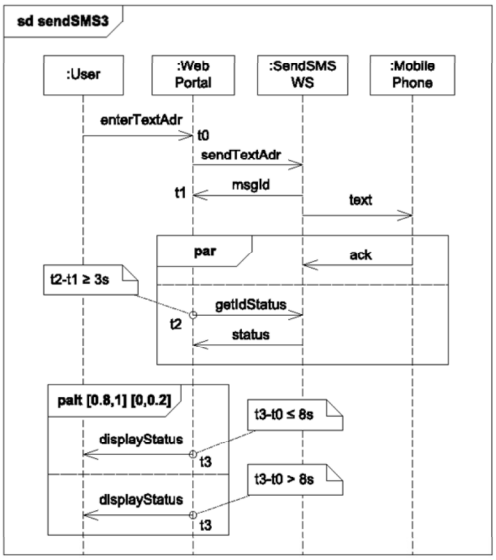
Distribution (Y + motivation) /N : N

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

Very clear, adheres strictly to the conventions of an UML class diagram.

Uses class names that are intuitively understood in the context of the domain.

Taken from:
Stepwise refinement of sequence diagrams with soft real-time constraints



Building blocks Which, what + (C | P):

- Life (time) lines (C) of communicating entities such a user (C | P depending on whether the user is human or another system), web-portal (C), a web-service SendSMSWS (C), mobile phone (P, according to intention a device I C, because strictly speaking the diagram shows an object)
- Subdiagrams (C) for concurrent or alternative scenarios
- Comments (C) containing real-time timing constraints

Connectors Which, what + (C | P):

- Labelled messages (C), whose endpoints (sending or receiving event) may be labeled with a moment in time

View – concern – stakeholder (1..*):

1. Process view, scenario interesting for system integrators and testers who must see whether the relative timing of events is ok and have an interest in concurrency (par block). Also interesting to users and acquirers of the system to see whether the system is performant enough. If the user is another system developers of that system may use the timing information to resubmits queries based on time-out.

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

- Security: N
- Availability & reliability: N
- Maintainability: N

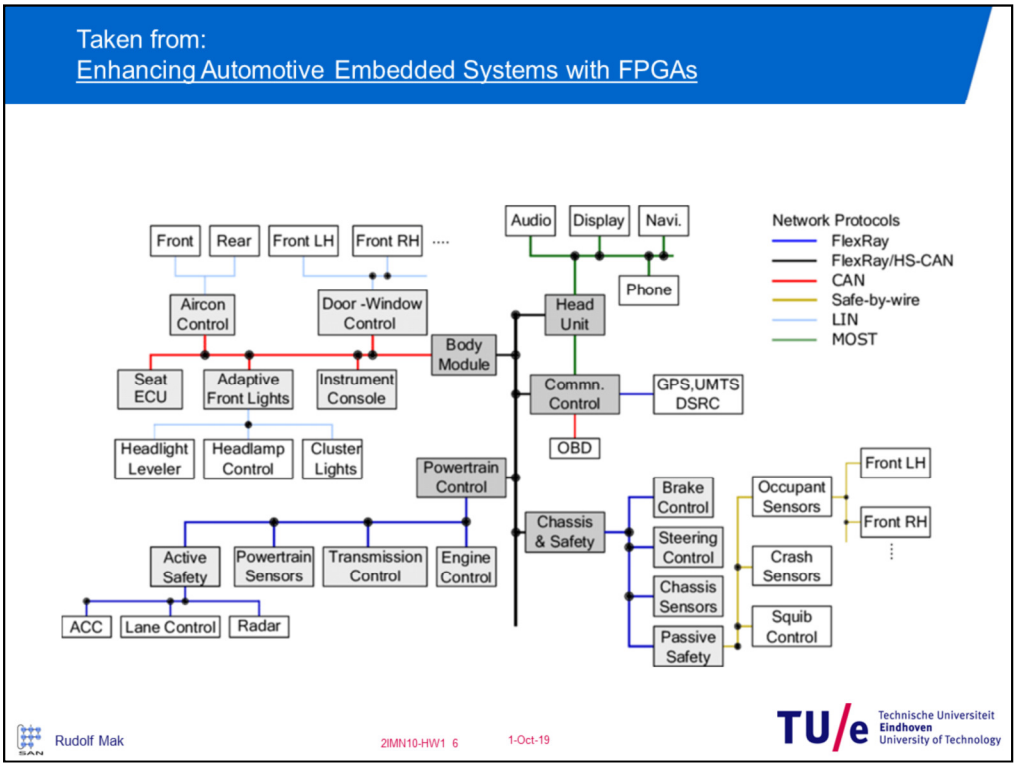
- Performance and scalability: Y,
    - We see timing constraints and concurrency

Distribution (Y + motivation) | N: Y,

- Because the user, the web-portal and mobile phone are distinct entities and presumably at different locations. Moreover, there will be many users of the web-portal and many mobile phones reachable from the portal through the web-service.

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

- Classical UML sequence diagram with extra element.
    - The probabilistic alternative, explained in the paper from which the diagram is taken, otherwise unclear.
- Message names clumsy “sendTextAdr” instead of “Send(text, addr)” which would make clear that the user specifies two things.



Building blocks Which, what + (C | P): all P

- A variety of devices in a car:
  - Control units (light grey boxes)
  - Sensors and activators (most white boxes)
  - Multimedia devices or subsystems
  - Gateways (dark grey boxes) connecting subsystems using a different protocols
- Legend with protocols (C)

Connectors Which, what + (C | P):

- Various networks (P) classified according to protocol (C)

View – concern – stakeholder (1..*):

1. Logical view. Of interest to stakeholders that need an overview of all subsystems in the car, such as system engineers, but also to stakeholders that need to know the car's capabilities, e.g. the vendor and marketing (multimedia capabilities), and to assessors of car safety that need to see which subsystems make driving the car save, but also whether subsystems can contain propagation of software faults or cannot otherwise interfere.
2. Physical view. Of interest to system engineers that have to assemble the system. In particular, the gateways that interface the various network protocols are of interest to them.

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

- Security: N
- Availability & reliability: Y,
    - Subsystems on separate networks offer isolation and fault containment (see reference)
- Maintainability: N
- Performance and scalability: N

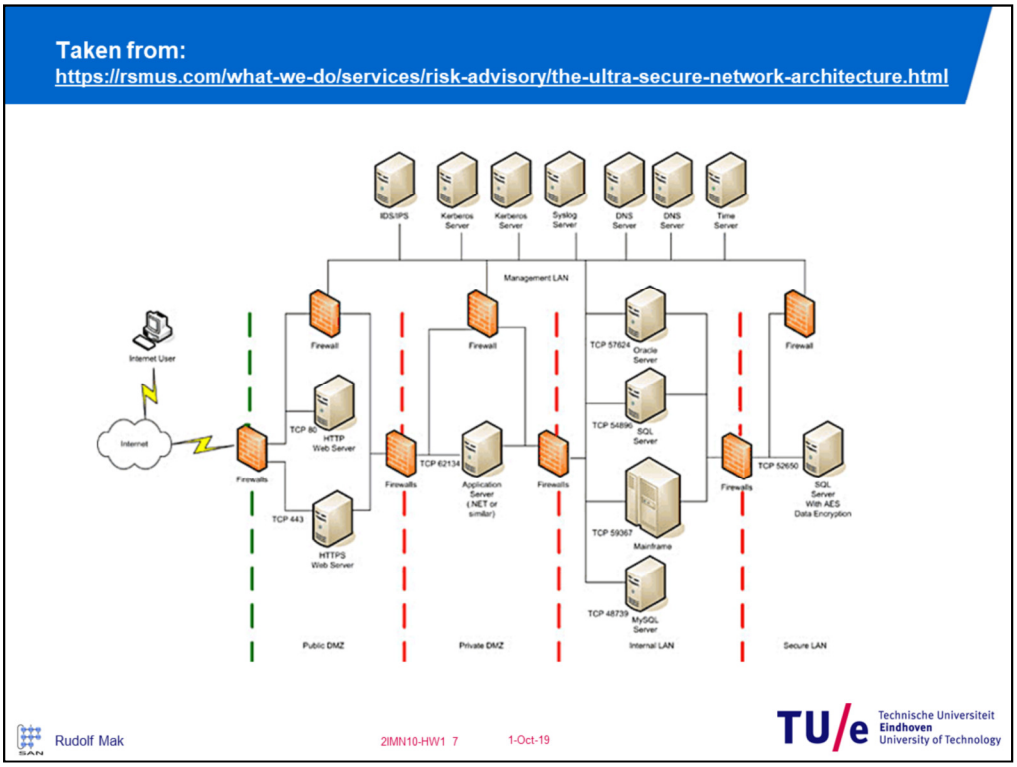
Other: Safety, both exhibiting devices to provide passenger safety in case of driving accidents, but also to prevent the car itself to cause/become a safety threat.

Distribution (Y + motivation) | N: Y,

- the system consists of many networked devices that can independently fail

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

- Distinction between gateway and control not always clear



Building blocks Which, what + (C | P):

- Servers (P) of all kind,
- Mainframe (P),
- Firewall (C | P) either a dedicated device (e.g. a router/gateway) configured as such, or a process running on such a device.
- Internet (C)
- Security zones (parts of the LAN shielded by firewalls): demilitarized zones (DMZ), internal and secure LAN
- User (C)

Connectors Which, what + (C | P):

- LAN network segments (C|P) occasionally labelled with protocols and service end points, TCP … (C)

View – concern – stakeholder (1..*):

1. Physical view of interest to system engineers who have to put the system together. Also system owners/users can see that their data cannot be accessed by unauthorized parties.
2. Operational view of interest to operators and system administrators that have to configure and maintain the security infrastructure. The model is also of interest to acquirerers and other stakeholders that need to assess the security status of the system

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

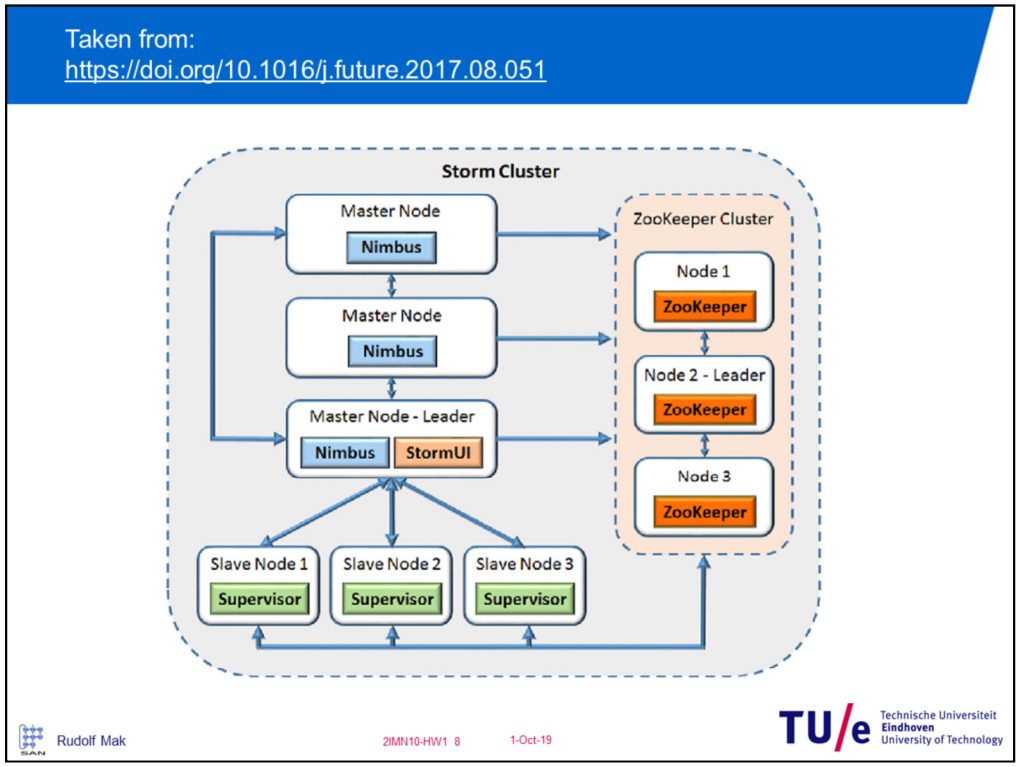
- Security: Y,
    - Because we see firewalls used to create DMZs, Kerberos servers for authentication, and extra secure SQL-server for encryption
- Availability & reliability: N
- Maintainability: N
- Performance and scalability: N

Distribution (Y + motivation) | N: Y

- Because, by definition a LAN consists of many different networked machines

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

- Clearly indicates the location where firewalls need to be inserted.



Building blocks Which, what + (C | P):

- Storm cluster (P)
- Zookeeper subcluster (P)
- Nodes organized in a cluster (P): masters, slaves, leader
- Processes (deamons) (C) running on nodes: Nimbus, Zookeeper, Supervisor
- Monitor process (C) StormUI

Connectors Which, what + (C | P):

- Communication channels (C) between nodes. One-way between Zookeeper nodes and Master nodes (a mistake?)

View – concern – stakeholder (1..*):

1. Deployment view interesting to system engineers and operators that have to install and configure the system. All stakeholders that need to have a general overview of the runtime platform of the Strom framework for stream processing, e.g. end-users.

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

- Security: N
- Availability & reliability: Y,
    - Redundant master and Zookeeper node for fault tolerance

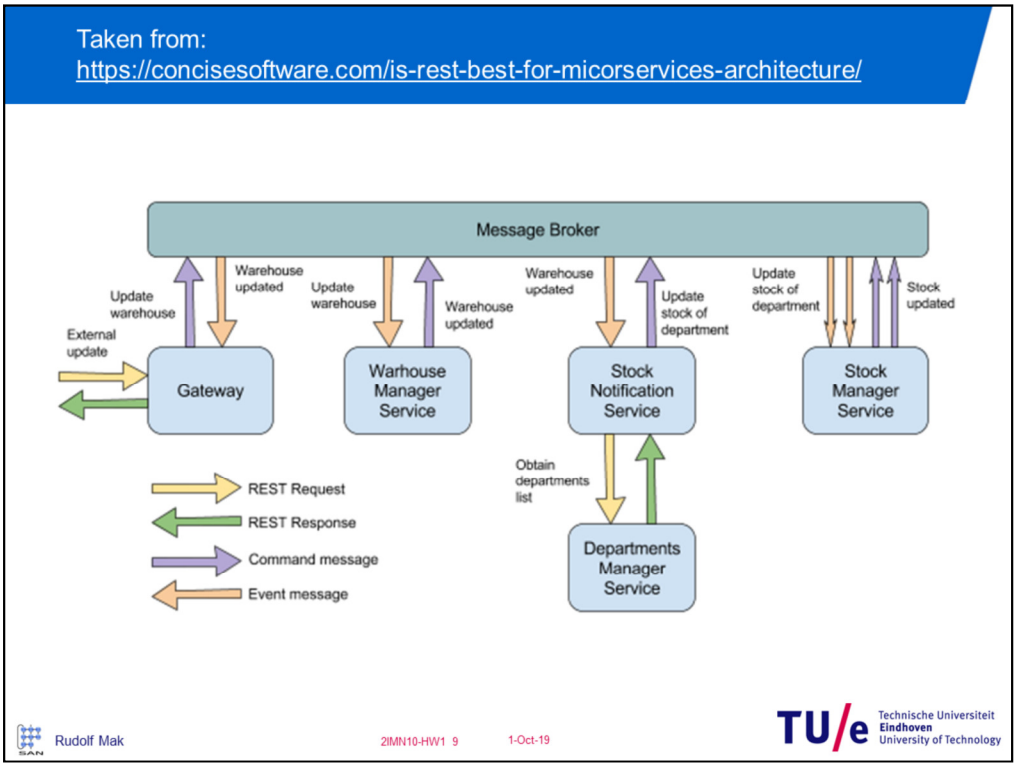
- Maintainability: N
- Performance and scalability: Y,
    - Slave nodes are there to perform tasks, coordinated by the local Supervisor component. Thus we have concurrency and horizontal scalability by increasing the number of slave nodes.

Distribution (Y + motivation) | N: Y

- We see redundant master nodes and Zookeeper nodes for fault tolerance. We multiple slave nodes for performance

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

- Semantics are difficult to understand without basic knowledge of cluster computing and knowing what zookeeper does. Otherwise the diagram is clear.



Building blocks Which, what + (C | P):

- Services (C)
- Gateway (C) translates REST requests into message messages
- Message broker (C; can also be seen as a connector)

Connectors Which, what + (C | P):

- Messages, containing commands (C)
- Message broker (C; can also be seen as a building block)

View – concern – stakeholder (1..*):

1. Process view: By identical naming of commands and event messages, control flow is indicated, albeit rather implicit (could be considered a scenario). Interesting for system integrators.

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

- Security: N,
    - In principle the gateway can deal with security issues, but there is no indication of that found in the diagram
- Availability & reliability: N
- Maintainability: Y,
    - because the services are decoupled and binding is via a message broker. Hence they can, e.g., be independently be updated. Also the system can easily be extended with new

services.

- Performance and scalability: Y,
    - Eventing and notification tend to minimize data transfer. Data is only transferred to interested parties. Lack of concurrency can be observed by sending multiple update commands to a single Stock Manager Service. Replicating this service would provide concurrency.

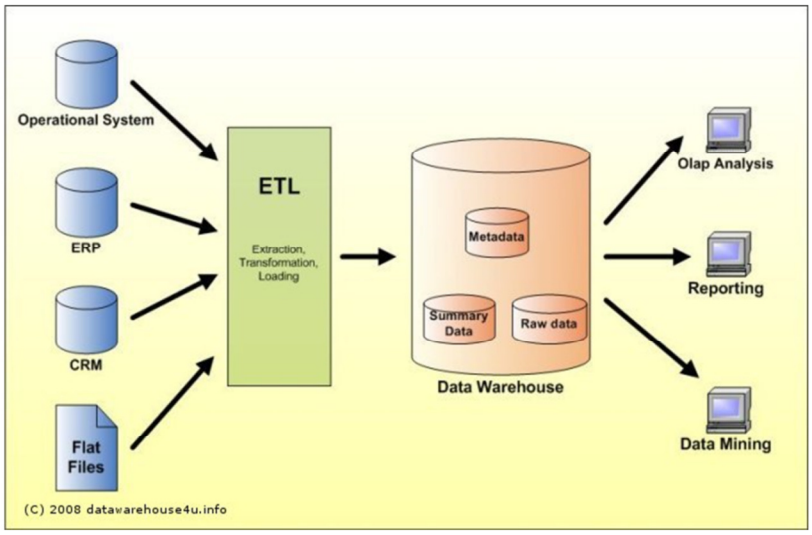
Distribution (Y + motivation) | N: N

- In principle the services could be distributed and reside on different hosts in the company network, but all services could equally well reside on a single machine (they are microservices!). A deployment view is needed to decide this.

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

- The diagram does not exhibit the workflow explicitly as it is explained in the linked document. Semantics of the update is not clear, since it is not clear which data is communicated between and kept by the individual services. My guess: “Warehouse has products in stock for multiple departments (shops?). Upon a change in the warehouse content all departments that sell updated information kept by the (Departments Manager Service) need to be notified. In addition, a nasty spelling mistake “war”house instead of “ware”house.

Taken from:
https://www.datawarehouse4u.info/



Building blocks Which, what + (C | P): All C

- (sub)Systems: the ETL system, and Data Warehouse, and various input systems
- Multiple user applications (on the right-hand side) that process the warehouse data:
- Multiple data sources (on the left-hand side): databases, files, system (ERP, CRM, OS)
- Individual data stores within the warehouse

Connectors Which, what + (C | P): All C

- Data streams

View – concern – stakeholder (1..*):

1. Context view (RW), because it shows the systems an ETL system or a Data Warehouse interacts with. Interesting to stakeholders that have to explain the fundamentals of “data warehousing”, such as educators, or that have to document such a system, but not for stakeholders that want to use, acquire, build, … a real system and are probably aware of this. In the Kruchten viewpoint library this would be best classified as belonging to the logical view.

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

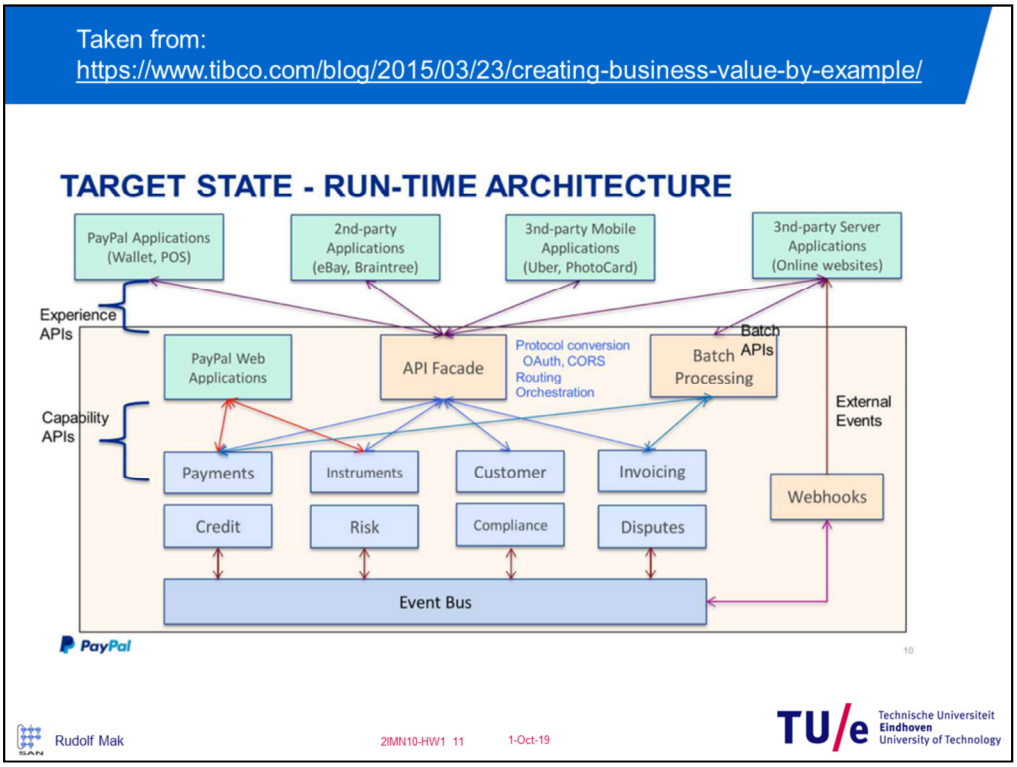
- Security: N
- Availability & reliability: N
- Maintainability: N
- Performance and scalability: N

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

- A data warehouse usually contains vast amounts of data in multiple data bases, in all likelihood stored on many machines spread over one or more datacenters, but the model hardly shows that.

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

- Clear because only the essentials are shown. However, is it intended to show the context of the Warehouse or of the ETL system, or both? Either way, not a very exiting diagram!



Building blocks Which, what + (C | P):

- System: yellow box (C)
- APIs (C) exposed to the system environment (yellow boxes)
- Applications (C) both from PayPal itself and part of the system or
- Capability APIs (C) the blue boxes (organized in two layers which may be irrelevant) and not directly accessible to none PayPal apps.
- Event bus (C): communication subsystem for eventing
- Façade (C)

Connectors Which, what + (C | P): all C

- Uses or exposes-to relationship between external application and the yellow system APIs (the open APIs)
- Delegation/forwarding relationships between the publically exposed APIs and the internal ones
- Events between the Event bus and capability APIs. Not all capability APIs are connected to the bus. It is not clear whether this is done yo avoid clutter in the diagram or whether there is indeed a distinction between APIs in this respect

View – concern – stakeholder (1..*):

1. Development view, because it shows the organization of APIs (services). Interesting to programmers.
2. Context view. We see the system in its environment, and how other companies may use Paypal

services to build their own business application. Within the other companies these stakeholders can be the company owners but also developers within that company that have to build the applications. For the owners/board of PayPal itself, the diagram shows how business value can be increased by exposing services to 3rd parties.

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

- Security: Y,
    - because the façade offers authentication
- Availability & reliability: N
- Maintainability: N
- Performance and scalability: N

Distribution (Y + motivation) | N: N

- Although the scale of PayPal, and the mentioned third parties that use it, suggests that there must be many servers that expose the mentioned APIs

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

- Even with the text from the linked webpage and the video from InfoQ this is rather incomprehensible. Much of the above is based on educated guessing. Confuses the APIs with the services they expose. The curly braces are probably drawn in the wrong position (too high). What is an experience API (undefined concept)?