Architecture of Distributed Systems

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 (C) or physical (P)?
2. Same questions as 1, but now for connectors? Do not forget (C) or (P).
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  Which, what + (C | P): 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 Which, what + (C | P): 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 that indicating 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 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, because there are BBs for access control (ACList, BlackList)
• Availability & reliability: N
• Maintainability: Y, because the diagram 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.
Building blocks Which, what + (C |P):
- DataCenters (P), Storage nodes (P), containing the MySQL database (C), backup storage (P) with distributed file system (C)
- Processes (C): the application, routers (application level processes not devices!), snapshot services, cluster manager (Helix), coordination service(Zookeeper), data replicators, databusses, data analysis (Hadoop)
  - because the databusses and cluster manager are identified by name they can also be considered as (P)

Connectors Which, what + (C | P):
- Bidirectional and unidirectional data flow (C), labelled with protocol information (C)
- Control messages, realizing a workflow on the cluster (C)

View – concern – stakeholder (1..*):
- Logical: global overview of Linked-Ins expresso system, intended for all stakeholders to get a general understanding of the system
- Control and dataflow would be interesting in process view but to abstract to really be useful.
• Similarly, too little info to be really useful for a deployment view, even though we see both processes and physical entities.
• Process view, we see concurrency, task and data distributed over cluster nodes and coordinated by Zookeeper and scheduled by Helix.

Extra-functional requirements (Y + motivation) | N :
• Security: N
• Availability & reliability: Y, because we see backup
• Maintainability: N
• Performance and scalability: Y,
  we see replication of data and processes to allow concurrent processing. Cluster allows resource capacity to be adapted to application load, hence also scalability

Distribution (Y + motivation) | N : Y
• Multiple nodes in multiple datacenters, multiple copies of a process running within a datacenter

Clarity/Semantics (☺ | ☺ | ☹) + motivation: ☺,
Drawing well organized, but nodes that run processes are missing.
Needs a lot of additional explanation to know how the system operates.
Building blocks Which, what + (C |P):
Actors (C): Devop, User
Machines (P): Master node, Slave odes, Docker registry
Processes (C): apiserver, scheduler, replication controller, kubelet, pod, proxy
Command line interface (C): kubectl (not possible to infer from diagram or text at link)

Connectors Which, what + (C |P):
Manages dependency: dashed line between kubelet and pods (C)
Solid lines indicate communication

View – concern – stakeholder (1..*):
• Logical: for all stakeholders a general overview of the kubernetes system, for its users to see how to set up their applications.
• Not much use for developers of the kubernetes system

Extra-functional requirements (Y + motivation) | N :
• Security: N
• Availability & reliability: Y, replication promotes availability and reliability
• Maintainability: N
• Performance and scalability: Y, because pods are replicated on demand in response to increased load, but it is difficult to see this from the diagram. The application runs on multiple nodes.

Distribution (Y + motivation) | N : Y, we see multiple nodes, pods are sets of docks that net to be collocated.

Clarity/Semantics ( ☺ | ☺ | ☺ ) + motivation: ☺,
From the diagram it is impossible to determine what the purpose (workflow orchestration and resource management) of the Kubernetes system is.
Building blocks Which, what + (C | P): all (C)
- Activities, the rounded squares
- Swimlanes (vertical strokes) with timed activities (the rounded boxes with text).
  - Each swimlane is labeled by an actor (Player) or component (Client, Internet, Server) that performs the actions.

Connectors Which, what + (C | P): all (C)
- Event (mouse click between player and client),
- Messages between client, internet and server.

View – concern – stakeholder (1..*):
- A scenario, from which players can determine the responsiveness of the game. As such, it also belongs to the user view.
- Process view, because its specifies the nature of the interaction between components. Interesting to testers that can verify the timing requirements.

Extra-functional requirements (Y + motivation) | N:
- Security: N
• Availability & reliability: N, although a system experiencing high latency may appear to a user as (temporarily) not available, this is usually not considered to be included under availability.
• Maintainability: N
• Performance and scalability: Y,
  • Performance, because the actions are provided with timing information that indicate response time for Players.
  • Scalability can be seen to be poor, because the server is state-full, forwards the “world update” to all players.

Distribution (Y + motivation) | N : Y, there are at least a client and a server.

Clarity/Semantics (☺ | ☺ | ☺) + motivation: ☺,
Follows the conventions of an UML activity diagram closely.
Building blocks Which, what + (C | P):
  - Layers (not strictly layered) (C)
    - Autosar RTE (C) (red layer, probably defines the interface)
    - Applications, services (C)
    - Drivers (C)
    - Microcontrollers (P)
  - Error sources (C + P, see legend)
  - Libraries (cross cutting) (C)

Connectors Which, what + (C | P):
  - Network connection between ECU (electronic Control Unit) (P)
  - Calling sequences (C)

View – concern – stakeholder (1..*):
  - Development view
    - Developers are interested in the organization of RT-functionality in modules and layers
  - Physical view
    - (Doubtful although) some physical element are identified, in particular as error sources
    - Interesting for system engineers
  - Assessors can determine from both views how the system deals with errors
  - Both views are of interest stakeholders that have a safety concern. Although the focus of the
model is on error sources, it gives an overview of the system which is of use for all stakeholders that want to understand the system (that concern could be attributed to a logical view)

Extra-functional requirements (Y + motivation) | N :
• Security: N
• Availability & reliability: Y, because it is indicated that E2E protection detects interference errors
• Maintainability: Y, because the system is partitioned into modules grouped in layers according, thereby reducing dependencies
• Performance and scalability: N

Distribution (Y + motivation) /N : Y
Although only two ECUs are shown, these stand for an arbitrary pair of the many ECUs that can be present in a modern car.

Clarity/Semantics (😊 | ☹️ | ☹️) + motivation: 😊
Intention of diagram is clear, due to the legend. Indication of content of the modules a little vague.
Building blocks Which, what + (C | P):
- Actors (Developer, Administrator, Consumer of a web-site) all (C, roles)
- Use cases (C)
- Package cloud (C)

Connectors Which, what + (C | P):
- Labelled associations (develop, Administer, Visit) that show for each actor in which use case it is involved (C)
- Labelled relationships (probably include, possibly extend) between use cases (C)
- Grouping of use case within a package Cloud (C)

View – concern – stakeholder (1..*):
- Functional (logical view), because it shows the various users, the functionality they may expect from the system
- Deployment view, because it indicates that the system for maintaining and building is deployed in the cloud

Note that this is about a system for building, maintaining and using a cloud-based
web-site.
The developer and administrator actors pertain to the web-sites built with the system not the system itself.
Hence, this diagram does neither belong to the development nor to operational view of the system.

Extra-functional requirements (Y + motivation) | N :
• Security: N
• Availability & reliability: Y, because creation of backups and adding resources creates reliable and available web-sites.
• Maintainability: N
• Performance and scalability: Y, there are use cases for increasing the resources of a web-site, which can easily be done when it is hosted in the cloud

Distribution (Y + motivation) | N : N, see remark after views.

Clarity/Semantics (☺ | ☻ | ☹) + motivation: ☻,
Sloppy usage of use diagram conventions, in particular the connectors. Labels on the connectors could as well be modelled as specific use cases.
Not clear whether the relationships between use case is “include”
Building blocks  Which, what + (C | P):  
- Layers (C)  
- Devices (all P): servers at the higher layer (could be conceptual but the suggestion is dedicated machines),  
  Human machine interfaces, controllers, PLC, RTU at lower layers,  
- Remote users (C)  
- Switches, routers (P)  
- Legend indicating the type of layer

Connectors Which, what + (C | P):  
- Internet (C)  
- Firewalls (C) (could also be seen as a building block)  
- Networks/busses (P): internal to (LAN) and in between layers

View – concern – stakeholder (1..*):  
- Development view: Shows how functionality is distributed over the layers, with clear separation of concerns. Interesting for programmers and others that have to realize, install, configure this functionality.
• Deployment view: shows how the functionality is allocated to devices in various layers of abstraction, and where security elements have to be inserted in the network connections. Interesting for system engineers, and system administrators that build and operate the system.
• Both views are interesting for assessors to see what security measures and mechanism are envisioned.

Extra-functional requirements (Y + motivation) | N :
• Security: Y, we see firewalls controlling the traffic between layers, DMZs (demilitarized zones, a security pattern for access control), authentication servers
• Availability & reliability: N
• Maintainability: Y, organization in layers helps maintainability by reducing dependencies, in particular for managing security
• Performance and scalability: N

Distribution (Y + motivation) | N : Y, each layer contains a lot of machines.

Clarity/Semantics (☺ | ☺ | ☺) + motivation: ☺,
Clear drawing, legend helps to determine the semantics.
Building blocks  Which, what + (C | P): (all C)
- processes such Client, Trackers, and processes that operate on client data (Mapper, Shuffle, Reducer)
- storage objects (HDFS, Local)
- lifelines and activations, repetition blocks

Connectors Which, what + (C | P): (all C)
- asynchronous messages (start, launch)
- synchronous messages with return values (read, write)
- Creation events (start, launch)

View – concern – stakeholder (1..*):
- Process view, interesting for testers and integrators that have an interest in putting the system together and verifying that the parts interact in a proper fashion
- Also a scenario (of a complete map-reduce computation), as such it also shows interaction of the user with the system, which is part of the logical view.

Extra-functional requirements (Y + motivation) /N :
- Security: N
- Availability & reliability: Y,
  - Implicitly, because Hadoop inherently addresses this issues.
  - Explicitly, because we see a “heartbeat” message showing the TaskTracker is alive.
• Maintainability: N
• Performance and scalability: Y
  • Concurrency is visible through messages that start threads and nodes, and launch subcomputations

Distribution (Y + motivation) /N : N, not through the diagram. Of course, it is well-known that Hadoop is a framework deployed on clusters.

Clarity/Semantics (☺ | ☻ | ☼) + motivation: ☻
• Sequence diagram that follows the UML conventions.
Building blocks Which, what + (C |P):
• Files and packages, (P) since referred to by name.

Connectors Which, what + (C |P):
• Containment (“consists of” or “includes”) relationships (C)

View – concern – stakeholder (1..*):
• Development view, organization of game software into packages, containing sets of files of interest to programmers for development and maintenance

Extra-functional requirements (Y + motivation) | N :
• Security: N
• Availability & reliability: N
• Maintainability: Y, because it shows the organization of the software, which helps locate files that need e.g. modification or bug repair.
• Performance and scalability: N

Distribution (Y + motivation) | N: N
Clarity/Semantics (☺ | ☺ | ☺) + motivation: ☺
Little information, well displayed.
Building blocks Which, what + (C|P): all (C)
- Discovery service (C)
- Requesters/Providers: (humans and agents) grouped into an entity
- Documents/information (C) regarding the service (FD, WSD, Sem), as labels of interactions

Connectors Which, what + (C |P): all (C)
- Service description collection between provider and discovery service (1a)
- Service discovery: request (1b) with criteria, response (1d) with WSD
- Semantics negotiation (2), between human requester and provider
- Agent configuration with service details (3)
- Service usage interaction (4)

View – concern – stakeholder (1..*):
- Process view, interesting to service providers and consumers to see which steps need to be taken, and in which order, to discover and use a service.
- Development view, because it shows the documents required for discovery and usage of a service. Interesting for developers and users that have to create these
documents.

• Logical view: Since the description is generic this is also interesting for educators/communicators that have to explain SOA style.

Extra-functional requirements (Y + motivation) | N :
• Security: N
• Availability & reliability: Y??

In principle, availability discusses the readiness for service of an existing service, but one could argue that the discovery procedure shows how a service becomes available.
• Maintainability: Y, it tells providers (and consumers) of services which documents they have to draw up, so that a service can be used.
• Performance and scalability: N

Distribution (Y + motivation) | N : Y, we see at least three logical parties.
• Note that the discovery service may actually be taken care of by either the requester entity or the provider entity

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

To fully understand the interactions, especially 2 and 3, the accompanying text is needed. Abbreviations could have been explained in a legend.