Technische Universiteit
Eindhoven
University of Technology

Architecture of Distributed Systems (2IMN10)

Homework Assignment 1

Matej Grobelnik (1281674)
Bram van de Wall (0911838)
Keyshav Suresh Mor (1237978)

September 21, 2018

1 VICSDA

1.1 Building blocks

The building blocks are Services(C), Service Management(C) , Credit Management(C), Certificate Management(C), Repository(P), Recommender(P) and Orchestrator(P).

1.2 Connectors

All (C): Part-of relationships, Is_a relationships (to indicate special services), Dependency relationships

1.3 View – concern – stakeholder

The model is through an information viewpoint which concerns with storing, manipulating and information distribution. It concerns users, acquirers, developers and maintainers.

1.4 Extra-functional requirements

- Security: No
- Availability & reliability: No
- Maintainability: Yes, class structure is shown at a level useful for implementation and modification. An overview for developer of the structure of the program is given.
- Performance and scalability: No

1.5 Distribution

Yes there is a concept of distribution in the model as the repository, services, credits & services management, recommendation policy etc. are all separately organised and interact to provide the service.

1.6 Clarity/Semantics

The clarity and semantics of the model are very good and clearly explains the interaction of users with the services and internal mechanism.

2 Espresso (distributed document store)

2.1 Building blocks

- Data centers (P)
  - Online Data Center
  - Offline Data Center
  - Remote Data Center
- Storage Nodes (C — P)
- Software components (C — P) (P as components may have dedicated hardware allocated to them)
  - Application
  - Router
  - Databus
  - Data Replicator
  - Snapshot Service
  - Helix
- Storage Engines (C)
  - DFS
  - MySQL
  - ZooKeeper

2.2 Connectors

- Data Flow (C)
  - HTTP
  - MySQL Replication
  - MySQL Backup/Restore
- Control Flow (C)
  - ZooKeeper protocol
- Containment
  - Part of data center (P)
  - Uses storage engine (C)

2.3 View – concern – stakeholder

- Logical view: The end users in this case are application developers. What they can see in this diagram is that the application they develop will only need to communicate with the router.
- Physical view: Not as detailed, but we can at least see the per data center grouping.
- Process view: System integrators and testers can see how the various parts interact. We can also see how components are scaled independent of each other.

2.4 Extra-functional requirements

- Security: N
- Availability & reliability: Y, replication and backup/restore
- Maintainability: N
- Performance and scalability: Y, storage nodes are horizontally scalable

2.5 Distribution

Y, Many different systems communicating with each-other. Even with completely different data centers.

2.6 Clarity/Semantics

+, gives a good overview of how the system is put together. Dis not give unnecessary details such as how data is routed or backups are made/restored.

3 Kubernetes

3.1 Building blocks

Actors (C): Devops, User, API server (P)
Cluster nodes (C), Infrastructure container(C), Proxy (C), Docker Registry (P), Scheduler (C), Controller (C), Containers (P)

3.2 Connectors

- Data Flow (C)
  - HTTP
  - Internet
- Dependency relationship (C)

3.3 View – concern – stakeholder

Process view: System integrator can see how the various components communicate with each other.

3.4 Extra-functional requirements

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

3.5 Distribution

Probably yes, since it has many components not necessary on one system.

3.6 Clarity/Semantics

:— Diagram gives us simple overview but all connections are not clear.

4 MMOG-RTT

4.1 Building blocks

Actions (C)
Columns(C): Player(P), Client (C), Internet (C) , Server (P).

4.2 Connectors

The connectors between Player and Client, Internet and Server are Physical whereas between Client and Internet are conceptual.

4.3 View – concern – stakeholder

The viewpoint presented is a functional viewpoint highlighting functionalities, interfaces and interactions.

4.4 Extra-functional requirements

Performance & Scalability : No (Latency in performance)
Availability/Reliability :
Security : No
Maintainability : Yes because of easy uncomplicated architecture

4.5 Distribution

Distribution between player, clients and server side.

4.6 Clarity/Semantics

The layout and design is pretty self explanatory and semantics are easy to understand.

5 AUTOSAR

5.1 Building blocks

- Applications (C)
  - OS-Application 1
  - OS-Application 2
- Libraries (C)
- Runtime Environment Layer (C)
- OS Services (C)
  - System Services
  - Memory Services
  - Communication Services
- Hardware Abstraction (C)
  - Onboard Device Abstraction
  - Memory Hardware Abstraction
  - Communication on Hardware Abstraction
  - I/O Hardware Abstraction
  - CDD
- Drivers (C)
  - Microcontroller Drivers
  - Memory Drivers
  - Communication Drivers
  - I/O Drivers
- Microcontrollers (P)
  - Microcontroller 1 / ECU 1
  - Microcontroller 2 / ECU 2
- Senders / Receivers (C)
- Error/Interference sources (C)

5.2 Connectors

- Messaging (C—P): Arrows between senders and receivers. Physical in the case of external microcontrollers.
- Containment (C): Senders / Receivers inside of other components
- Vertical Alignment (C): Layering, what parts can talk to what

5.3 View – concern – stakeholder

- Development view: Developers of the OS can see the different layers and how different components relate to them.
- Logical view: Application Developers, which are the end users here, can see the places that errors returned by function calls they make originate from.
- Physical view: The mapping of software components to hardware is depicted.

5.4 Extra-functional requirements

- Security: N
- Availability & reliability: Y, various typical error sources are described.
- Maintainability: N
- Performance and scalability: N

5.5 Distribution

Y, communication with other Microcontrollers.

5.6 Clarity/Semantics

Neutral. It does give a reasonably clear example where errors can happen during communication, but it is not clear what happens after an error has occurred. Neither is it really clear how the various parts of the OS relate to each other.

6 Use-Case diagram

6.1 Building blocks

Use cases (C), Cloud (C)
Actors (C): Developer, Administrator, Consumer

6.2 Connectors

Associations between actors and use cases (C)

6.3 View – concern – stakeholder

Logical view: We can clearly see use cases interact between other use cases and actors. Useful for system acquirers and developers.

6.4 Extra-functional requirements

- Security: N
- Availability & reliability: N
- Maintainability: Yes, it shows use cases for creating, saving, cloning, maintaining web instances.
- Performance and scalability: N

6.5 Distribution

Yes, it represents cloud environment which by its definition is a distributed system.

6.6 Clarity/Semantics

:) It is clear, understandable, and simple use case diagram.

7 Network Topology

7.1 Building blocks

The building blocks are Bus Network (P), Controller LAN (P), Supervisory LAN (C), Operation DMZ (P), Enterprise LAN (P) , Internet DMZ (P), Firewall (C), Servers (C), Layers (C), Remote users (C), routers (C), HMI (C), Controllers (C).

7.2 Connectors

Layer 0 to 1 : Physical
Layer 1 to 2 : Conceptual
Layer 2 to 3 : Physical
Layer 3 to 4 : Physical
Layer 4 to 5 : Physical
Layer 5 to Internet : Conceptual

7.3 View – concern – stakeholder

The system shown is in Context viewpoint and is relevant to all stakeholders like acquirers, users and developers.

7.4 Extra-functional requirements

Performance/Stability : Y because of the application specific server access
Availability/Reliability : Yes because of separate channels for separate applications
Security : Y because of firewalling of requests.
Maintainability : Y because of compartmentalised architecture.

7.5 Distribution

There is a distribution between the Internet, the servers and the input devices.

7.6 Clarity/Semantics

The clarity and semantics of the architecture are well described ans easy to understand.

8 Hadoop MapReduce

8.1 Building blocks

- Lifelines (C)
- Two Loops (C)

8.2 Connectors

- Containment (Loops) (C)
- Method invocation (C)

8.3 View – concern – stakeholder

- Process view: System integrators and testers can see how the various parts interact.

8.4 Extra-functional requirements

- Security: N
- Availability & reliability: N (Though, heartbeat messages are part of the diagram, error conditions are not)
- Maintainability: N
- Performance and scalability: Y, tasks are spread over various nodes

8.5 Distribution

Y, Communication with nodes to spread the work over many different machines.

8.6 Clarity/Semantics

+, gives a good overview of how data/calls flow through a mapreduce cluster. Does not provide information about error conditions

9 Python game package hierarchy

9.1 Building blocks

Software package (C), software sub-packages (C), files (C)

9.2 Connectors

File hierarchy (C)

9.3 View – concern – stakeholder

Development view: The structure of the software package is visualized, such that the programmer has a clear view of how the package is organized.

9.4 Extra-functional requirements

- Security: N
- Availability & reliability: N
- Maintainability: Yes, it shows how package is organized.
- Performance and scalability: N

9.5 Distribution

No.

9.6 Clarity/Semantics

:) Gives clear view over the package structure.

10 SOA

10.1 Building blocks

- Exchanged Data (C)
  - WSD
  - Sem
  - FD
  - Criteria
- Actors (C)
  - Requester Human
  - Requester Entity
- Entities (C)
  - Requester Entity
  - Provider Entity
- Agents (C)
  - Requester Agent
  - Provider Agent
- Discovery Service (C)

10.2 Connectors

- Discovery Communication (C)
  - Provider registration
  - Discovery request
  - Discovery request
  - Service handshake
  - Configure Requester Agent
  - Configure Provider Agent
- Agent Communication (C)
- Containment (C)
  - Requester Entity
  - Provider Entity

10.3 View – concern – stakeholder

- Process view: System integrators can see how the various components communicate with eachother.

10.4 Extra-functional requirements

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

10.5 Distribution

Y, communication between various components

10.6 Clarity/Semantics

Negative, this could be represented using a standard UML diagram. It is also not really clear what the different line styles mean, or what is communicated exactly in what order.