SOFTWARE ARCHITECTURES

ARCHITECTURAL STYLES
SCALING UP PERFORMANCE
ARCHITECTURES

SW Architectures usually complex

Often we reduce the abstraction

Architectural Styles

• Layered style

Architectural Patterns

• Model View Controller
ARCHITECTURE STYLES

Basic Characteristics

Quality attributes
ARCHITECTURE STYLES

Data centric
• Databases

Call and return
• Part of this course

Implicit invocation
• Events

Independent components
• Peer to peer

Virtual Machines

Pipe and Filter - data flow
OVERVIEW

- **Domain and context model**
- **Arch. styles**
- **Reference architecture**

  - Small steps towards SW Architecture

Diagram:
- Domain and context model
  - Reference Architecture
    - Software Architecture
      - System Architecture
  - Architectural style
ARCHITECTURE
STYLES

Data centric

- Databases
- Voice recognition
- Compilers
ARCHITECTURE STYLES

Call and return

• OOD
• Procedural
• RPC
• AOP
• Layers
ARCHITECTURE STYLES

Call and return

- OOD
- Procedural
- RPC
- AOP
- Layers
ARCHITECTURE STYLES

Call and return

- OOD
- Procedural
- RPC
- AOP
- Layers
ARCHITECTURE STYLES

Call and return

• OOD
• Procedural
• RPC
• AOP
• Layers

Source code of methods

Security  Method logic  Synchronization  Logging

OOP

OOP + AOP

Aspects
ARCHITECTURE STYLES

Call and return

- OOD
- Procedural
- RPC
- AOP
- Layers
ARCHITECTURE STYLES

Implicit invocation

- Events
ARCHITECTURE
STYLES

Implicit invocation

• Events
ARCHITECTURE STYLES

Independent components
ARCHITECTURE STYLES

Virtual machines

<table>
<thead>
<tr>
<th>User Program 1</th>
<th>User Program n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System 1</td>
<td>Operating System n</td>
</tr>
<tr>
<td>Virtual 370</td>
<td>Virtual 370</td>
</tr>
</tbody>
</table>

VM370

Hardware

VIRTUAL MACHINE ARCHITECTURE (VM370)
ARCHITECTURE
STYLES

Pipes and Filters

Batch Sequential

Data Transformation

Classical data processing

Software Architectures

Data Flow
ARCHITECTURE STYLES

Pipes and Filters

Kinds of Data Flow Systems

In general, data can flow in arbitrary patterns.

Here we are primarily interested in nearly-linear data flow systems.

or in very simple, highly constrained cyclic structures.
ARCHITECTURE STYLES

Data centric
- Data integration, Distribution, Control, Coordination
- Scalability, Low coupling, Centralization, Reuse, Modifiable,

Call and return
- Modifiable, Reusable, Inf. hiding, Structural decomposition, Separation of concerns

Implicit invocation
- Modifiable, Low coupling, Hard to comprehend,

Independent components
- Integration, Scalability, Reuse, Low coupling, Distribution, Reliability

Virtual Machines
- Simulation, Emulation, Portability!, Flexibility, Lowered Performance, Extended features

Pipe and Filter
- Modifiable, Reuse, Easy design, Simplicity, Low Coupling,
- Slow, No filter cooperation, Lot of parsing
SCALING PERFORMANCE

Usual approach is to deploy app to a web server and provide access through HTTP/S

- Client-server architecture
  - Inside 3-layers and data repository
SCALING PERFORMANCE

Usual approach is to deploy app to a web server and provide access through HTTP/S

- Client-server architecture
  - Inside 3-layers and data repository
Minimize volume

ORM talks too much when not being careful

Usually HTTP (JS/CSS)
DEPLOYMENT, MAINTENANCE AND REPORTS

Get user experience

- Register your app at Google Analytics to see how it performs
- HTML5 allows you to see the statistics

Tomas Cerny, Software Engineering, FEE, CTU in Prague, 2016

<table>
<thead>
<tr>
<th>Name</th>
<th>Method</th>
<th>Status</th>
<th>Type</th>
<th>Initiator</th>
<th>Size Content</th>
<th>Time</th>
<th>Timeline</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.idnes.cz">www.idnes.cz</a></td>
<td>GET</td>
<td>200</td>
<td>text/html</td>
<td>Other</td>
<td>32.9 KB</td>
<td>164 ms</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>121 KB</td>
<td>162 ms</td>
<td></td>
</tr>
<tr>
<td>uni.css?r=043</td>
<td>GET</td>
<td>200</td>
<td>text/css</td>
<td><a href="http://www.idnes.cz">www.idnes.cz</a>:...</td>
<td>1.8 KB</td>
<td>78 ms</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Parser</td>
<td>3.1 KB</td>
<td>75 ms</td>
<td></td>
</tr>
<tr>
<td>gidnes.cz/css/idn3</td>
<td>GET</td>
<td>200</td>
<td>text/css</td>
<td><a href="http://www.idnes.cz">www.idnes.cz</a>:...</td>
<td>1.6 KB</td>
<td>83 ms</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Parser</td>
<td>3.0 KB</td>
<td>79 ms</td>
<td></td>
</tr>
<tr>
<td>reklama.css?r=043</td>
<td>GET</td>
<td>200</td>
<td>text/css</td>
<td><a href="http://www.idnes.cz">www.idnes.cz</a>:...</td>
<td>12.4 KB</td>
<td>384 ms</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Parser</td>
<td>36.3 KB</td>
<td>380 ms</td>
<td></td>
</tr>
<tr>
<td>portal.css?r=043</td>
<td>GET</td>
<td>200</td>
<td>text/css</td>
<td><a href="http://www.idnes.cz">www.idnes.cz</a>:...</td>
<td>11.1 KB</td>
<td>330 ms</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Parser</td>
<td>37.9 KB</td>
<td>326 ms</td>
<td></td>
</tr>
<tr>
<td>sph.css?r=043</td>
<td>GET</td>
<td>200</td>
<td>text/css</td>
<td><a href="http://www.idnes.cz">www.idnes.cz</a>:...</td>
<td>12.4 KB</td>
<td>384 ms</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Parser</td>
<td>36.3 KB</td>
<td>380 ms</td>
<td></td>
</tr>
<tr>
<td>uni.js?r=066</td>
<td>GET</td>
<td>200</td>
<td>application</td>
<td><a href="http://www.idnes.cz">www.idnes.cz</a>:...</td>
<td>42.4 KB</td>
<td>486 ms</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Parser</td>
<td>75.0 KB</td>
<td>480 ms</td>
<td></td>
</tr>
<tr>
<td>gidnes.cz/js/un</td>
<td>GET</td>
<td>200</td>
<td>application</td>
<td><a href="http://www.idnes.cz">www.idnes.cz</a>:...</td>
<td>2.0 KB</td>
<td>83 ms</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Parser</td>
<td>3.7 KB</td>
<td>79 ms</td>
<td></td>
</tr>
</tbody>
</table>

191 requests | 1.5 MB transferred | 4.65 s (load: 4.01 s, DOMContentLoaded: 3.62 s)
CLIENT-SERVER ARCHITECTURE

Properties:
• Centralization
• Easy with security
• Easy to locate
• Easy to scale
  • Until we reach the limit
  • Server is the bottleneck
• Performance influenced by the network conditions
  • And virtual distance between client and server
• Server has given throughput
  • Given by HW, our Design, Efficiency, Caching, etc.
CLIENT-SERVER ARCHITECTURE

Client

Server
CLIENT-SERVER ARCHITECTURE

Server throughput 300 clients at once

Client

Server
CLIENT-SERVER ARCHITECTURE

Client
20..300 peak

Server

Through put 300 clients at once

Load grows!

(a) Provisioning for peak load
CLIENT-SERVER ARCHITECTURE

Client
20..350 peak

Load grows!

Server

Through put 300 clients at once

Throughput 300 clients at once

(b) Underprovisioning 1
CLIENT-SERVER ARCHITECTURE

How to improve?

• Caching
• Performance analysis – profiling
• Native/Custom SQL queries for reports
• Better Hardware, more CPU/Mem

Client
20..350 peak

Server
How to improve?

- What if it is not enough?
- Indirection?
CLIENT-SERVER ARCHITECTURE

Client
20..350 peak

• Indirection?

Dispatcher

Server

How to improve?
CLIENT-DISPATCHER-SERVER ARCHITECTURE

Client
20..350 peak

• Indirection?

Dispatcher

Server

How to improve?

Tomas Cerny, Software Engineering, FEE, CTU in Prague, 2016
CLIENT-DISPATCHER-SERVER ARCHITECTURE

How to improve?

Client 20..350 peak

• Indirection?

Dispatcher

Server 1

Server 2
CLIENT-DISPATCHER-SERVER ARCHITECTURE

Client
20..350 peak

• Indirection?

Dispatcher

Server 1

Server 2

How to improve?
CLIENT-DISPATCHER-SERVER ARCHITECTURE

1) Pick a worker to forward request
   - Random
   - Round robin
   - Least busy
   - Sticky session / cookies
   - By request parameters
2) Wait for its response
3) Forward the response to client

Worker Pool

Client → Dispatcher → Worker

Tomas Cerny, Software Engineering, FEE, CTU in Prague, 2016
CLIENT-DISPATCHER-SERVER ARCHITECTURE

Most likely we cannot expect to multiply the throughput of the single server

- Balancing overhead
- We can balance different resources
  - Static vs. Dynamic
- Geo-location balancing
  - Content-Delivery-Network (CDN)
    - Static content (Akamai)
CONTENT DELIVERY NETWORK (CDN)

Example
CONTENT DELIVERY NETWORK (CDN)

Example
CONTENT DELIVERY NETWORK

Example
CONTENT DELIVERY NETWORK

Example
CONTENT DELIVERY NETWORK

Example
CONTENT DELIVERY NETWORK

Example
SCALING PERFORMANCE

Database might be the bottleneck

Database replication
SCALING PERFORMANCE

Database might be the bottleneck

Datagrid
SCALING PERFORMANCE

JBoss view on Datagrid
SCALING PERFORMANCE

JBoss view on Datagrid
SCALING
PERFORMANCE

JBoss view on Datagrid
SERVICE-ORIENTED ARCHITECTURE (SOA)

So far we considered that server-side app offers data, knowledge and presentation

Service does not provide presentation

Well accepted format

Standard: JSON, SOAP, XML...
Motivation
SERVICE-ORIENTED ARCHITECTURE

1960 - 1980
- Organization Focus
- Mainframe Centric
- Internal Use
- Unique Data

1990 - 2000
- Process Focus
- Client Server
- Partial Connectivity
- EDI File Transfer

2010 - 2050
- Distributed Functions
- Data Centric
- Universal Interoperability
- Real-time Connectivity
SERVICE

- Loose coupling
- Reusable
- Stateless
- Autonomous (independent)
- Discoverable
- Abstract
- Composable
- Platform independent
ANATOMY OF A SERVICE

Service Consumer

Interface Proxy

Service Interface

Service Implementation

New Service

Wrapped Legacy

Composite Service
SERVICES COMMUNICATE WITH MESSAGES

Providing reliability and security to messages
Sending messages across consumers and producers
Service Orchestration
BASIC WEB SERVICES

UDDI Registry

Points to description

Finds Service

Points to service

WSDL

Describes Service

Web Service Client
(J2EE, .NET, PL/SQL …)

Invokes with XML Messages

SOAP

Web Service
(J2EE, PL/SQL, .NET, C/C++, Legacy …)
ENTERPRISE SERVICE BUS (ESB)

It is a software architecture model used for designing and implementing the interaction and communication between mutually interacting software applications in service-oriented architecture (SOA).

- Model for distributed computing
- Variant of client server software architecture model
- Promotes flexibility with regards to communication & interaction between applications.
- Primary use in enterprise application integration (EAI) of heterogeneous and complex landscapes.
ENTERPRISE SERVICE BUS
From JBoss ESB Documentation
SOA IS AN EVOLUTIONARY STEP
SOA IS AN EVOLUTIONARY STEP

in distributed communications

hub and spoke

“too centralized”

EAI

point to point

“too decentralized”

Project-ware

bus

“just right”

SOA
TO ENABLE BUSINESS PROCESS OPTIMIZATION AND THE REAL TIME ENTERPRISE (RTE)

**Seamless End to End Process**

**BPM Expressed in terms of Services Provided/Consumed**

**Service to Customers**

- **Smart Clients**
- **Stores POS**
- **Mobile**
- **3rd Party Agents**
- **Portal**

**Service from Multiple Suppliers**

- **Internal Systems**

**SOA Patterns**

- **Single, Multi-Channel**
- **Service for consistency**

**SOA Pattern**: Standardized Service provided by multiple suppliers

Tomas Cerny, Software Engineering, FEE, CTU in Prague, 2016
Business functionality is duplicated in each application that requires it.

EAI ‘leverage’ application silos with the drawback of data and function redundancy.
SERVICE CENTRIC

Business scope

SOA structures the business and its systems as a set of capabilities that are offered as Services, organized into a Service Architecture.

Multiple Service Consumers
Multiple Business Processes

Multiple Discrete Resources
Multiple Service Providers

Service virtualizes how that capability is performed, and where and by whom the resources are provided, enabling multiple providers and consumers to participate together in shared business activities.
SERVICE CENTRIC APPROACHES

Open your business to extension and evolution!

Natural extension and reuse
• Expedia API, Paypal, Amazon API, Airfare, Heureka..

Open your system to novel needs, requirements, interaction
Reuse by other vendors!