Service Oriented Architectures
Motivation and Concepts

Johan Lukkien
System Architecture & Networking
Department of Mathematics & Computer Science
Eindhoven University of Technology
Just in case you thought...

- This presentation is *not* about web-services
- (though I might mention it occasionally)
Agenda

- Motivating examples
- SOA what?
  - some views from the web
- SOA elements and mechanisms
- Outlook, research targets
Example: classical embedded real-time system

- Network options
  - I: just a connected device/system
  - II: remote sensing/control (could combine with e.g. fieldbus technology in practice)
  - III: remote monitoring and control
Example: in-home networks

- Cassandra project within Philips research, in collaboration with CANDELA (ITEA), ISHARE (BSIK Freeband)

- Devices share
  - functionality
    - e.g. media processing functions
  - resources
    - e.g. database, computational platform
  - content
    - e.g. video

- Accessssible as networked services ("service units")
CASSANDRA distributed and decentralized content analysis in Connected Home / Planet

Home Network Topologie
Distributed clients (CE devices)

Meta DB
Content DB

SU OFL
pnx1300

SU Overlay by text detection
pnx1300

SU Film mode identification
pnx1300

SU Face detection
pnx1300

SU Object tracking
pnx1300

SU Audio/Music

SU TOG

SU Chaptering

SU Movie

SU AV Jukebox

SU AV Genre classification

SU Automatic Speech Recognition
GPP x86

SU Face recognition
GPP x86

Analog / digital AV stream

Rendering device A

Rendering device B

Rendering device C

From CDB

From CDB

From CDB

From CDB
Example: integration of business applications

Applications and (end-)services

Canonical Information Model

Customer
Business Transaction
Employee

Business System 1
Business System 2
Business System 3
Business System 4
Microsoft: “Entity aggregation”

- Single view on an entity
- Horizontal partitions
  - service composition
- Queries across services
  - combine data & metadata

(MSDN article “SOA Challenges Entity Aggregation”)
Points of view on S.O.A.

1. **Portfolio view** [Business environment]
   - stand-alone applications with overlapping functionality
     - data-storage (name, address, .....)
   - .... need to:
     - have a unified view on data (Microsoft: ‘entities’)
     - build new functionality by combining existing applications (flexibility, agility)

2. **Solution view** [Embedded, networked environment]
   - packaged functionality
     - processing, storage, content
   - applications realized through unforeseen cooperations
     - e.g. consumer electronics
   - heterogeneity (platform, connectivity), mobility
   - need for embedded ‘decision taking’
Essentially,

1. A systematic approach to the existing situation
   - disparate applications
   - an uncontrollable evolution process
   - the wish to quickly and easily realize new applications *with existing material*.

2. The need to have effective plug-in and coordination support
   - external *policies* while the *mechanisms* are exposed by the (services inside the) devices

   • ... similar issues, differences in granularity, goals
Agenda

- Motivating examples
- SOA what?
  - some views from the web
- SOA elements and mechanisms
- Outlook, research targets
A service-oriented architecture (SOA)

- defines how two computing entities interact in such a way as to enable one entity to perform a unit of work on behalf of another entity.
- The unit of work is referred to as a service,
- the service interactions are defined using a description language.
- Each interaction is self-contained and loosely coupled, so that each interaction is independent of any other interaction.
• A service-oriented architecture is
  – a way of connecting applications across a network
  – via a common communications protocol.
• In theory, this lets developers treat applications as network services that can be chained together to create a complex business process more quickly.
• SOA is
  – an architectural style
  – whose goal is to achieve loose coupling among interacting software agents.

• A service is
  – a unit of work done by a service provider
  – to achieve desired end results for a service consumer.
  – Both provider and consumer are roles played by software agents on behalf of their owners.

Dr. Hao He
Wikipedia.org

• **Service-oriented architecture (SOA)** is
  – a software architectural concept
  – that defines the use of services to support business requirements.

• In an SOA,
  – resources are made available to other participants in the network as independent services
  – that are accessed in a standardized way.

• **Most definitions of SOA**
  – identify the use of web-services (using SOAP and WDSL) in its implementation,
  – however it is possible to implement SOA using any service-based technology.
Some other comments

• Sounds like: food-oriented lunch
• It appears to be about quickly changing contacts between components
• Not new....
  – where did CORBA go?
  – what about GRID?
  – toolbus, service bus – seem to address the same issues
Overloaded acronym.....

- Save Our Animals
- School Of The Americas
- School Of the Americas
- Semiconductor Optical Amplifier
- Server Of Authority
- Service Order Administration
- Service Oriented Architecture
- Ships On The Air
- Society Of Assassins
- Soldiers Of Agony
- Sons Of Aiur
- Source Of Authority
- Southern Africa Fund, Inc.
- Special Operations Aircraft
- Spring Over Axle
- Start Of Authoritative
- Start Of Authority
- State Oceanic Administration
- State Of Affairs
- Statement Of Activity
- Stimulus Onset Asynchrony
- Student Organization Accounts
- Subaru Of America
- Sulphate Of Ammonia
- School Of Assassins
CBSA

- Focus on component platform
  - docking ....
- Platform dependent
  - registration & lookup
  - language bindings, machine architecture
- Little awareness in the application of some non-functional aspects
  - mobility
  - distribution (often transparent)
  - security
  - timing, cost
- S.O.A. represents some sort of a next step
Example: Robocop Component Architecture

• ITEA projects ROBOCOP, Space4U
S.O.A. features (requirements)

- Interoperability
- Loose coupling
  - clear interfaces and dependencies
  - late binding
    - even at run-time:
      - advertisement, discovery
      - based on descriptions
      - avoidance of language, OS, ISA binding
- Composability
- Location transparency

- ..... network-exposed services
  - ‘network as system bus’
Drivers

- Ambient intelligence
  - how do we expect (spontaneous, automatic) cooperation of these many embedded devices?
    - optimizing towards user experience
    - optimizing towards resource use
    - not knowing each-other upon design time

- Re-use
  - at deployment level
  - need to deploy components such that they can be integrated in new configurations as such, at run-time
    - including support for non-functional properties

What is needed such that future as yet unforeseen cooperations are possible?
Service: working definition

- **Service**: a contractually specified overall functionality (semantics) of an object.
- **Service quality**: non-functional properties of a service (e.g. speed, reliability, ...).
- **Service interface (API)**: actions (“primitives”) and responses that make the service available; these responses can be autonomous (“events”, “call-backs”). In addition, a specification that
  - describes their effect on state variables and parameters, as well as their results;
  - describes rules as how and in what sequence to call them;
  - describes the functional and non-functional properties of sequences of calls.
  (i.e., the interaction or access protocol)
- **Service access point**: location where the service is accessed
Simplified view on running component

- Four dependencies
- Single processor
  application: 3, 4
- Client: 1, 3, 4
- Server: 2, 4
- (In fact: GUI can be a separate component)

- Example: DNS
Interoperability views

Interoperability focused on protocol; no language or platform binding besides message structure and semantics.
Agenda

- Motivating examples
- SOA what?
  - some views from the web
- SOA elements and mechanisms
- Outlook, research targets
S.O.A. elements

• Roles
  – service provider, service consumer
    • note: a service is not a component; it may be exposed by a component
    – perhaps: broker (service), manager / controller (consumer)

• Services
  – interfaces
  – descriptions (schema’s)
    • semantics
    • access points

• Interaction
  – discovery, advertisement
  – access: control & eventing

• Structural control (‘orchestration’)
  – setting up service compositions

• Protocols
Model

Service

ServiceImplementation

ServiceAdvertisement

ServiceConsumer

ServiceDescription

This relationship is built around Eventing and Control mechanisms of Service Oriented Architecture

This relationship is focused on the service discovery mechanism

“publish-find-bind-execute”
Sample technologies

• Let’s practice some acronyms
• Web-services
  – UDDI, WSDL, SOAP, XML
• Universal Plug’n Play
  – SSDP, GENA, SOAP, XML
• GRID technology
• Technologies to build S.O.A.
  – CORBA, DCOM
  • though these are much more tightly coupled
  – MOM, e.g. IBM MQ systems
Purpose of service discovery

- Locate appropriate service provider
  - “distributed query processing”
- Obtain details about service interface and quality
- Obtain service access point (address of)
- Decide upon (negotiation) interaction details
  - security, other qualities
  - access rights, payment, ...
Taxonomy of service discovery

- Pre-configured
  - Location aware
  - Mediated
    - Non-transparent
    - Transparent
  - Immediate
    - Active
    - Passive

from “JESA Service Discovery Protocol” by Stephan Preuss, Networking 2002, LNCS 2345
Examples

• SSDP (simple service discovery protocol)
  – non-configured, immediate, active
    • both provider (advertisement) and consumer (discovery)
    • note: either one passive gives polling inefficiency

• UDDI (Universal Description, Discovery and Integration)
  – configured (?), mediated, non-transparent

• SLP (Service Location Protocol)
  – non-configured, mediated, non-transparent & immediate passive (service)
Control Mechanisms

- Typically: from service consumer to service
- Mainly: remote calls

<table>
<thead>
<tr>
<th>Security</th>
<th>Access control</th>
<th>Synchronization</th>
<th>Intermediation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trusted calls</td>
<td>Exclusive</td>
<td>Synchronous</td>
<td>Direct</td>
</tr>
<tr>
<td>Trusted callers</td>
<td>Concurrent</td>
<td>Asynchronous</td>
<td>Mediated</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Transparent</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Non-transparent</td>
</tr>
</tbody>
</table>

1/31/2006
Examples

- SOAP/HTTP
  - synchronous, non-transparent mediation / immediate
- Corba, RMI
  - synchronous, immediate
  - Corba: also mediated via broker
Eventing mechanisms

- Typically: from service to service consumer
  - can be realized as a “reverse service”
- Covers ‘data driven’ interactions
- Similar issues as for control

<table>
<thead>
<tr>
<th>Initiation</th>
<th>Poll</th>
<th>Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security</td>
<td>Trusted calls</td>
<td>Trusted callers</td>
</tr>
<tr>
<td>Intermediation</td>
<td>End to end</td>
<td>Relayed/mediated</td>
</tr>
<tr>
<td></td>
<td>Single mediator</td>
<td>Multiple mediators</td>
</tr>
<tr>
<td></td>
<td>Implicit</td>
<td>Functionally equivalent mediators</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Heterogeneous mediators</td>
</tr>
<tr>
<td>Timing</td>
<td>Real-time</td>
<td>Virtual-time</td>
</tr>
<tr>
<td>Event handling</td>
<td>Blocking</td>
<td>Non-blocking</td>
</tr>
<tr>
<td>Transport</td>
<td>Guaranteed delivery</td>
<td>No guaranteed delivery</td>
</tr>
</tbody>
</table>
Examples

• GENA/HTTP
  – interrupt, non-blocking, virtual time, guaranteed delivery

• Real-time traffic/RTP
  – (interrupt), blocking, real-time, non-guaranteed
Service Oriented Applications

- **Serviceoriented.org**: “Service oriented applications are composed of the run-time loose bundling of services. It is said that these services are orchestrated to solve some problem.”

- Set up a ‘graph’ of collaborating services
  - e.g. connect a content providing service to a player
- ... or just another layer – (broker-like)
  - given a collection of services found on the network....
  - ....realize a new service
  - ... as in the Entity Aggregation view
Orchestration

- System topology
  - hierarchical (layered)
  - arbitrary

- Service – service connections
  - mediated
    - only via controller
  - direct
    - controller can set up connections, typically using the event mechanisms
Usage patterns

• Reconciliation and aggregation
  – see earlier examples: different data formats

• Partitioned data
  – combine similar data from different sources
    • e.g. UK & Dutch locations of Royal Dutch Oil

• Augmentation
  – add meta-data obtained from analysis
    • e.g. the VCA application, using face recognition to enrich the video stream

• Distributed processing
  – different functions on same data, concurrent operations
    • e.g. the mentioned VCA-application
  – distributed resources
Examples

- BPEL: Business Process Execution Language – E.g.
  - Accept the name, postal address, and e-mail address of a user.
  - Look up the current weather forecast for the user by zip code.
  - Download information about the address the user has provided.
  - Send an e-mail with the collected data to the user's provided e-mail account.

- Research issue
Agenda

• Motivating examples
• SOA what?
  – some views from the web
• SOA elements and mechanisms
• Outlook, research targets
Debates and taxonomies

- Stateless / Statefull services
- Interaction styles
  - publish & subscribe
  - ‘forward’ call sequence
  - data and control flow
    - push / pull
- Granularity of services
- Hierarchical or flat control

- Major worry: granularity / performance balance!
Challenges

• Focussed service discovery
  – in combination with locality, qualities, ownership, membership, ...
  – ... just general queries

• Embedded decision procedures
  – interpretation of descriptions, selection, learning

• Security, privacy, ownership

• Evolution path
  – include legacy, even currently developed legacy
Challenges

• Service development
  – self-containedness, granularity, performance
  – interfaces exposing *mechanisms* for non-functional properties

• Application development
  – ‘language’ having S.O.A. elements as primitives
  – specify policies
  – deal with mobility and connection failure as *regular behavior*
    • time/space separation
  – exception handling
  – analysis: visualization / simulation
Conclusions

• S.O.A. represent a next step in component based software architectures
  – composition *after deployment*
  – cooperation *via the network*
  – explicit ‘*information faces*’ of components
  – application as *orchestration*

• Though perhaps not new, S.O.A. enforces focus on
  – thinking about components as part-of-a-whole
  – sharp semantic boundaries, including non-functional properties
    • just extending Parnas’ principle