Service Management Concepts and Challenges in Cloud Environments

Tihana Galinac Grbac
Software Engineering and Information Processing Laboratory - SEIPLab
Faculty of Engineering
University of Rijeka

COST ACROSS 2nd Summer School, May 27-31, Opatija, Croatia
Autonomous Control for Reliable Future Networks and Services
Literature

- Tankovic, N; Galinac Grbac, T; Truong, H-L.; Dustdar, S: Transforming vertical Web applications into Elastic Cloud Applications, Proceedings of International Conference on Cloud Engineering (IC2E 2015), 9-12 March, 2015, Phoenix, USA.
- Pradeeban Kathiravelu, Tihana Galinac Grbac, Luís, Veiga: Building Blocks of Mayan: Componentizing the eScience Workflows Through Software-Defined Service Composition, Accepted for ICWS 2016, San Francisco, USA. Nikola T.
- Galinac Grbac T., Runeson P, ’Plug-in’ Software Engineering Case Studies, CESI – ICSE 2016 Workshop, Austin, USA
Agenda

1. Motivation
2. Background
3. Service Orientation
4. Cloud Services and Network Function Virtualisation
5. Challenges
   - Service design performance aware
   - Service verification and certification
   - End-to-end service reliability and availability
6. Conclusion
Motivation

Complex system behaviour

As we analyse within EVOSOFT and ACROSS projects
Key problems with software evolution

- More and more software systems tend to evolve towards complex software systems (e.g. IoS) and systems of systems (SoS)
- Interconnection of peripheral systems over distributed network into system of systems (IoT)
Facts about Complex Software System

• Complex systems did not evolve accidently
• Huge effort is invested - there must be a great interest to grow into complex system
• Developed in sequence of projects over decades
• Mostly perform tasks that are
  – of crucial importance for community (defense, energy, public services, banking, health)
  – for very large number of end users (telecommunication)
• Quality is of crucial importance
Problems with complex systems?

- Key problems become:
  - How to manage complex software system?
  - Are we able just by observing properties of system parts to predict its overall behavior?
System requirements

• Parallel execution of multiple different requirements, for number of users
  – e.g. Systems implementing MSC logic has to cope with more than million requests in parallel,
  – provide number of different ‘standard’ protocol interactions
• high availability for its users
  – If certain malfunction happen the peers has to be timely informed, and all related resources properly released, avoid congestion situations
• properly dimensioned – avoid load
• response by the required time
  – Real time system, a system with a real-time constraints
• Interoperable with other vendors equipment
• Inside logic has to provide external protocol compliance
System requirements

• Parallel execution of multiple different requirements, for number of users
  – e.g. Systems implementing MSC logic has to cope with more than million requests in parallel,
  – provide number of different ‘standard’ protocol interactions
• high availability for its users
  – If certain malfunction happen the peers has to be timely informed, and all related resources properly released, avoid congestion situations
• properly dimensioned – avoid load
• response by the required time
  – Real time system, a system with a real-time constraints
• Interoperable with other vendors equipment
• Inside logic has to provide external protocol compliance
System requirements

• Easy to maintain
  – system divided into logical functions
  – Well defined and separated logical functions
  – Easy to trace system dynamics
  – Easy transformed from object code back to original code
Migration to virtualized environments

• Trend is to provide everything ‘as a service’
• Network is provided to its users ‘as a service’ by providing:
  – Infrastructure (processing, memory)
  – Operating platforms
  – Software applications
• Numerous users may get any network resource as a service and pay per use
Focus on ‘Service’

• Solution in new software abstractions, network management concepts
• Service orientation
Background
First switches

Switching software in telecommunication network

Picture taken from: http://commons.wikimedia.org/wiki/File:WAC_telephone_operators_operate_the_Victory_switchboard_during_the_Potsdam_Conference_in_their_headquarters_in...-_NARA__199007.jpg
Further evolution

- Evolution of information flow management:
  - 0 and 1 – Relay switch
  - Assembly languages - low-level programming language for a computer or other programmable device specific to a particular computer architecture
  - Algorithmic programming languages - executable program is sequence of algorithms
  - Object oriented programming languages – executable program is set of objects, dynamic
  - Service oriented software – services are self-contained elements, distributed in network that may dynamically form service chains to accomplish specific end user need
Service Orientation
Software Oriented Architecture

- An architectural style of building software applications that promotes loose coupling between components so that you can reuse them and work within a distributed systems architecture.

- Some SOA product has been built by many industrial well accepted frameworks but also as part of some virtualisation environments e.g. Mirantis.
Software specialised for specific function (Example: Signal denoising algorithm)

Software for specific functions is needed to accomplish functionalities of application software in different application domains (e.g. Knee analysis)

Application domain

Different kinds of terminals and end equipment

*Photos used from http://www.freedigitalphotos.net/
Definition of Service

- A service is an self contained entity that provides service to its clients by using interfaces and exchanging messaging.
- Message exchange provides stateful operation of service for its clients.

![Diagram of service interaction]

S1

Request

Response

S2
Web Services

• A industry standard
• Web Service-related standard
  – WSDL: describe WS
  – SOAP message: sent between WS
  – UDDI: register WS
• Web Service uses many kind of transport medium: HTTP, SMTP, JMS...
• Web Service can go through firewall easily
Web service model

- Service Registry
  - Publish or announce (WSDL)
  - Find or discover (UDDI)
  - Bind or invoke (SOAP)

- Service Provider
- Service Requestor
Benefits of service binding through Service Registry

- Service provider may
  - Dynamic change of web services during runtime
  - Perform CRUD operations (Create, Run, Update, Delete) @runtime
  - do not have to maintain track of Service users to perform service management
- Service requestor may
  - choose among number of services without explicitly knowing service address (binding)
  - may switch among service provides @runtime
- Cloud provider may develop recommender systems to secure justice and harmony for its users
  - Measurements of service, service provider’s and service consumer’s behaviour
- track record of list of available services via service registry
Cloud Services and Network Function Virtualisation

Standardisation
Cloud Services

• European Telecommunications Standards Institute (ETSI) launched the Cloud Standards Coordination (CSC) *
  – Summarize relevant standards for Cloud services to their users and service providers
  – Identify and Collects Cloud Service Use Cases

• Service Measurement initiatives:
  – Service Measurement Index
  – Cloud Services Measurement Initiative Consortium (CSMIC)

Network Function Virtualisation: ETSI GS NFV

- Aims to transform the way that network operators architect networks by evolving standard IT virtualisation technology to consolidate many network equipment types onto industry standard high volume servers, switches and storage, which could be located in a variety of NFVI-PoPs including datacentres, network nodes and in end user premises.

- Main objectives are following:
  - Rapid service innovation through software-based deployment and operationalization of network functions and end-to-end services.
  - Improved operational efficiencies resulting from common automation and operating procedures.
  - Reduced power usage achieved by migrating workloads and powering down unused hardware.
  - Standardized and open interfaces between network functions and their management entities so that such decoupled network elements can be provided by different players.
  - Greater flexibility in assigning VNFs to hardware.
  - Improved capital efficiencies compared with dedicated hardware implementations.

*Network Functions Virtualisation (NFV); Use Cases
http://www.etsi.org/deliver/etsi_gs/nfv/001_099/001/01.01.01_60/gs_nfv001v010101p.pdf
Network Function Virtualisation

- Virtualisation and Application Management
- Cloud API
- Cloud Stack
- OpenStack

Examples of Virtual Network Function:
- Switching: BNG, CG-NAT, routers.
- Mobile network nodes: HLR/HSS, MME, SGSN, GGSN/PDN-GW, RNC.
- Home routers and set top boxes.
- Tunnelling gateway elements.
- Traffic analysis: DPI.
- Signalling: SBCs, IMS.
- Network-wide functions: AAA servers, policy control.
- Application-level optimisation: CDNs, Load Balancers.
- Security functions: Firewalls, intrusion detection systems.

*Source: http://www.ietf.org/proceedings/88/slides/slides-88-opsawg-6.pdf*
NFV Orchestrator:
- deploying of new Net. Serv. (NS),
- NS lifecycle management performance measurements, event correlation, termination)
- global resource management, validation and authorization of NFVI resource requests
- policy management for NS inst.

VNF Manager:
- lifecycle management of VNF instances
- overall coordination and adaptation

Virtualized Infrastructure Manager
- controlling and managing the NF resource usage
- collection and forwarding of Perform. measurements and events


COST ACROSS 2nd Summer School, May 27-31, Opatija, Croatia
Autonomous Control for Reliable Future Networks and Services
MANO - VNF descriptor model

<table>
<thead>
<tr>
<th>Name</th>
<th>Cardinality</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VNFD_elements</td>
<td>1</td>
<td>This describes a set of elements related with the entire template (VNFD).</td>
</tr>
<tr>
<td>VNF_elements</td>
<td>1</td>
<td>This describes a set of elements related with a particular VNF instance.</td>
</tr>
<tr>
<td>VDU_elements</td>
<td>1...N</td>
<td>This describes a set of elements related to a particular VDU. Each VDU will have a set of its own elements.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Cardinality</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vendor</td>
<td>1</td>
<td>Specify the vendor generating this VNFD.</td>
</tr>
<tr>
<td>VNF_id</td>
<td>1</td>
<td>Specify the identifier (e.g. name)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Cardinality</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDU_id</td>
<td>1</td>
<td>A unique identifier of the said VDU, including version functional description and other identification information. This will be used to refer to VDU when defining relationships between them.</td>
</tr>
<tr>
<td>VM_specification</td>
<td>1</td>
<td>Provides a VM image or a reference.</td>
</tr>
<tr>
<td>Storage_req</td>
<td>0...1</td>
<td>Describes the required storage characteristics (e.g. size), including Key Quality Indicators (KQIs) for performance and reliability/availability.</td>
</tr>
<tr>
<td>Computation_req</td>
<td>0...1</td>
<td>Describes the required computation resources characteristics (e.g. processing power), including Key Quality Indicators (KQIs) for performance and reliability/availability.</td>
</tr>
<tr>
<td>Initiation</td>
<td>1</td>
<td>Defines the VDU initiation workflow including the functional script.</td>
</tr>
<tr>
<td>Termination</td>
<td>1</td>
<td>Defines the VDU termination workflow including the functional script.</td>
</tr>
<tr>
<td>Graceful_Shutdown</td>
<td>0...1</td>
<td>Defines the VDU graceful shutdown workflow (VDU is pre-warned and can take actions before the shutdown), including the functional script.</td>
</tr>
<tr>
<td>Other_constraints</td>
<td>0...1</td>
<td>Placeholder for other constraints.</td>
</tr>
<tr>
<td>High_availability</td>
<td>0...1</td>
<td>Defines redundancy model to ensure high availability examples include:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ActiveActive: Implies that two instance of the same VDU will co-exists with continuous data synchronization.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ActivePassive: Implies that two instance of the same VDU will co-exists without any data synchronization.</td>
</tr>
<tr>
<td>Scale_out/in</td>
<td>0...1</td>
<td>Defines minimum and maximum number of instances which can be created to support scale out/in.</td>
</tr>
</tbody>
</table>
Related literature for ETSI VNF and MANO

- Published E2E Arch, REQ, Use Case, Terminology documents in:
  - ETSI NFV Open Area:
    - http://docbox.etsi.org/ISG/NFV/Open/Published/
  - Published ETSI NFV white paper:
Free call service in Inteligent Network

- Service concept existing traditional telecommunication networks

So, what is novel?
Service/feature composition is well known problem in switching systems

*Book: 100 years of telephone Switching
http://books.google.ro/books?id=07NmhQoqwz&printsec=frontcover#v=onepage&q&f=false

*I. Lovrek, lectures, Faculty of Engineering and Computing, Zagreb,
Figure 12. Free call service in Inteligent Network
Cloud Services

• Cloud providers may be anybody
• Dynamic contracting among
  – Cloud providers
  – service provider and service user
  – Cloud providers and its users
• Service provider may be anybody, without exhaustive testing or certification software service
• Traditional telecommunication networks were developed by few development organisations driven by standards
• Exhaustive network testing of end user functions have proceeded before network use
virtualisation features (openStack)

• Automatic Scaling
• Load Balancing
• Service Orchestration
• Runtime CRUD operations
• Runtime Reconfiguration
• Service Chain and Service Group
Service Management Challenges
Management of Services

• We need to understand service and environment behaviour
• Services may be measured within the Cloud environment
• Service price and SA may be evaluated and compared among number of executed cases
• Entity behaviour may be determined from history:
  – Service (quality of service executions, popularity)
  – Service requester (his most favorite services)
  – Service provider (quality of his services)
Service Chain Management

• End user requirements may be realised through composition of services in service chains

• Dynamic management of each service may affect the service chain performances, quality of service
1. Service Design
- performance aspect -
What logic is implemented in Service? Where are the boundaries?

• **Service identification.**
  What is a service? What is the business functionality to be provided by a given service? What is the optimal granularity of the service?

• **Service location.**
  Where should a service be located within the enterprise?

• **Service domain definition.**
  How should services be grouped together into logical domains?

• **Service packaging.**
  How is existing functionality within legacy mainframe systems to be re-engineered or wrapped into reusable services?


COST ACROSS 2nd Summer School, May 27-31, Opatija, Croatia

Autonomous Control for Reliable Future Networks and Services
What logic is implemented in Service? Where are the boundaries?

- **Service orchestration.** How are composite services to be orchestrated?
- **Service routing.** How are requests from service consumers to be routed to the appropriate service and/or service domain?
- **Service governance.** How will the enterprise exercise governance processes to administer and maintain services?
- **Service messaging standards adoption.** How will the enterprise adopt a given standard consistently?

Service quality modeling and prediction

- **Case Study:** Superius ECR (Electronic Cash Register)
  - previously monolithic SaaS web application
  - Problem:
    - how to deploy these service w.r.t. elasticity?
  - balance between SLA and PAYG policies

We need a support system to guide design decisions for optimal deployment scale wise solution.
Example: Electronic Cash Register
PhD student Nikola Tanković

<table>
<thead>
<tr>
<th>Variant</th>
<th>( EG_1 )</th>
<th>( EG_2 )</th>
<th>( EG_3 )</th>
<th>( EG_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( D_1 )</td>
<td>IS, FS, DS, SS</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( D_2 )</td>
<td>IS</td>
<td>FS, DS, SS</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( D_3 )</td>
<td>IS</td>
<td>DS</td>
<td>FS, SS</td>
<td>-</td>
</tr>
<tr>
<td>( D_4 )</td>
<td>IS, FS</td>
<td>DS, SS</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( D_5 )</td>
<td>IS</td>
<td>FS</td>
<td>DS</td>
<td>SS</td>
</tr>
</tbody>
</table>

*Tankovic, N; Galinac Grbac, T; Truong, H-L.; Dustdar, S: Transforming vertical Web applications into Elastic Cloud Applications, IC2E 2015, USA.
Experiment Results

- Cost difference between deployment variants up to 32%
- Achieved 6% cost reduction by applying fine-grained elasticity

*Tankovic, N; Galinac Grbac, T; Truong, H-L.; Dustdar, S: Transforming vertical Web applications into Elastic Cloud Applications, IC2E 2015, USA.
Service Runtime
- Quality of Experience/Quality of Service-
System performance and Quality of service
Example system

- Properties of complex software systems:
  - Large scale > 3 millions Lines of code
  - Open to external inputs
  - Distributed
  - Concurrent
  - High interaction between parts
  - Evolutionary developed

Example of complex software system is telecommunication software
Network communication architecture

- Networks are defined and modeled at different abstraction levels
Case study: How design of system architecture may effect the End service performance

- Signalling network in mobile communication network
- We can consider as high priority data traffic network
- Developed according 3GPP standardisation body regulations
Signalling network evolution
Phase 1: Introduction of ATM transport and new BICC protocol

- BSS
  - RANAP
  - MTP3b
  - SAAL
  - ATM
  - PHY

- BSSAP
  - MTP3
  - MTP2
  - PHY

- MGW
  - ATM

- MSC Server
  - TDM
  - ATMA

- BICC
  - MTP3
  - MTP2
  - PHY

- GMSC Server
  - TDM
  - ISDN, PSTN, PLMN
  - ATMA

- ISUP
  - MTP3
  - MTP2
  - PHY

- UTRAN
Signalling network evolution
Phase 2: Splitting of network architecture - new GCP protocol
M/G/1 Model
Processor load

1. UMTS-GSM / monolit node/ ATM core
2. UMTS-GSM / distributed node/ ATM core
3. UMTS-GSM / distributed node / IP core
Mean service execution time

1. UMTS-GSM / monolit node/ ATM core
2. UMTS-GSM / distributed node/ ATM core
3. UMTS-GSM / distributed node / IP core
Typical communication pattern
-Service chains-
System reliability and Quality of service
Problem:

- System verification activities have secured system reliability
- How these system properties will be secured in terms of these dynamic systems?
- How can we predict and model system behaviour in such dynamic environment?
- Can we predict service composition behaviour from local properties of each service in composition?
System verification and reliability

- Number of levels of abstraction
- Global properties of system and local properties describing component behaviour
- Impossible to derive simple rules from local properties towards global properties*
A small number of modules contain most of the faults

System and system components
If a small number of modules contain most of the faults, then it is because these modules constitute most of the code size.
Rel n

% of acc. faults in ST

% of modules

Rel n+1

% of acc. faults in ST

% of modules

Rel n+2

% of acc. faults in ST

% of modules

Rel n+3

% of acc. faults in ST

% of modules

Rel n+4

% of acc. faults in ST

% of modules

COST ACROSS 2nd Summer School, May 27-31, Opatija, Croatia

Autonomous Control for Reliable Future Networks and Services
Results of analytical distributions fit

Nonlinear regression fit for Pareto, double Pareto, Weibull and Lognormal distribution
Results of all studies

Ranking the probability distributions with respect to their performance in the non-linear regression fitting of the empirical samples for the random variable counting the number of faults in a software module

<table>
<thead>
<tr>
<th>Rank</th>
<th>This study</th>
<th>Concas et al. [15]</th>
<th>Zhang [14]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Double Pareto</td>
<td>Yule–Simon</td>
<td>Weibull</td>
</tr>
<tr>
<td>2</td>
<td>Lognormal</td>
<td>Double Pareto</td>
<td>Pareto</td>
</tr>
<tr>
<td>3</td>
<td>Yule–Simon</td>
<td>Lognormal</td>
<td>–</td>
</tr>
<tr>
<td>4</td>
<td>Weibull</td>
<td>Weibull</td>
<td>–</td>
</tr>
<tr>
<td>5</td>
<td>Pareto</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>
Service certification
Managing Contracts

- Traditionally the end user services were few, and tested in a network that is build based on fixed and known contracts.
- The most commonly used methods for ensuring the correctness of a system are simulation and testing.
- Exhaustive for any reasonably complex system is impossible.
- Errors can sometimes occur only for specific execution sequences which are difficult if not impossible to reproduce or debug, making an exhaustive analysis necessary.
- **Cloud network** is introducing dynamic contracting at all layers – impossible to test all situations.
- A Service Agreement (SA) represents a binding agreement between the provider and customer of a cloud service.
How can we secure reliable operation of stateful service chains in Cloud

• Run time testing
  – Testing combination space is reduced with additional knowledge from the runtime environment
• Behavioural type theory encompasses concepts such as interfaces, communication protocols, contracts, and choreography.
• As structural principle for building reliable software systems
• Idea:
  – to codify the structure of communication to support the development of reliable communication-oriented software.
  – to encode as types the communication structure of modern computer systems and statically verify behavioural properties about them
Example – Session types

• Aim: to develop programming languages, tools for development of certified software solutions for global services
• Developed language: e.g. Scribble for specifying network protocols

![Diagram of Example - Session types](image-url)

- (a) sequence diagram
- (b) behavioural type
- (c) transition system
Service Management in SDN network
QoS Aware service compositions

- Adaptive execution of scientific workflows
- More efficient and diverse service composition
- A very large-scale reliable service composition

Open Source community
- Find and consume the current best-fit
- Among the multiple implementations or deployments of the same service.

*Pradeeban Kathiravelu, Tihana Galinac Grbac, Luís Veiga: Building Blocks of Mayan: Componentizing the eScience Workflows Through Software-Defined Service Composition, Accepted for ICWS 2016, San Francisco, USA.
QoS Aware service compositions

- Based on performance measurements collected in controller

*Pradeeban Kathiravelu, Tihana Galinac Grbac, Luís Veiga: Building Blocks of Mayan: Componentizing the eScience Workflows Through Software-Defined Service Composition, Accepted for ICWS 2016, San Francisco, USA.
QoS Aware service compositions

COST ACROSS 2nd Summer School, May 27-31, Opatija, Croatia

Autonomous Control for Reliable Future Networks and Services
Conclusion

• Service management challenges are originating from virtualizing execution environment and enabling dynamic change:
  – Change of software development paradigm from developing software for particular hardware to developing a generic service
  – Runtime adaptation mechanisms based on history behaviour (of services, cloud environment, etc.)
  – Need for extensive empirical studies, lot of empirical measurements, efficient analyses algorithms
  – New information management concepts that would hide private details but provide benefit for autonomous system control
  – Reliable autonomous systems have to solve challenges of service management
  – Importance of reliable autonomous systems are for cloud systems in mission critical domains such are energy networks, health care, automated home environments