The aim of the Leeds Beckett Repository is to provide open access to our research, as required by funder policies and permitted by publishers and copyright law.

The Leeds Beckett repository holds a wide range of publications, each of which has been checked for copyright and the relevant embargo period has been applied by the Research Services team.

We operate on a standard take-down policy. If you are the author or publisher of an output and you would like it removed from the repository, please contact us and we will investigate on a case-by-case basis.

Each thesis in the repository has been cleared where necessary by the author for third party copyright. If you would like a thesis to be removed from the repository or believe there is an issue with copyright, please contact us on openaccess@leedsbeckett.ac.uk and we will investigate on a case-by-case basis.
SOFTWARE ENGINEERING FOR SERVICE-ORIENTED ARCHITECTURE

Dr Muthu Ramachandran, FBCS, FHEA, Principal Lecturer, SETEP Research Group Lead, School of Computing, Creative Technologies, and Engineering (CCTE), Leeds Beckett University, Email: m.ramachandran@leedsbeckett.ac.uk, BCS EA SIG Conference, 17th July 2017, BCS London.
Service and cloud computing has revolutionized the way we develop software. The emergence of cloud computing has huge impact on the economics of developing services.

However, it remains challenging to understand the concept of a service which is quite new for computing sciences.

This talk will present a systematic approach to understanding service and cloud computing and will provide a Service Development Life Cycle approaches and market niche techniques and tools.
In this Lecture, discover

- Leeds Beckett University and Computing Research

  - Definition & Introduction
  - SOA Architecture & Applications
  - Cloud Computing

Part 2: Cloud Software Engineering vs Software Engineering for the Cloud
  - Cloud Service Requirements Engineering
  - Cloud Service Design
  - Cloud Service Development & Deployment
  - Cloud Service Testing

Part 3: SOA & Cloud Development Process & Standards

Reading List & References
Leeds Beckett University

- Leeds Beckett is the fifth best uni in the country for producing CEOs
- Beckett breeds more business leaders than Oxford, LSE or Leeds Uni
- The data, released by Emolument, studied 26,000 graduates across the UK – and found that Beckett produces 3.1 per cent of the UK’s CEOs, CTOs and Partners.
Research Groups at Leeds Beckett

- **Software Engineering, Technologies, and Emerging Practices (SETEP) Group (Lead by Muthu)**
- Assistive technologies and computer forensics and security
- IT and sustainability
- Data mining, big data and AI
Current & Some Previous Projects

• Software Engineering – Reuse Framework, Component Model for Complex Systems, used in various Industries including Philips, Image Systems, Volantis Systems Research, etc., SPI, Testing, Software Product Line Engineering, Maturity model for CBSE and SOA
• Software Security Engineering Research: Vulnerability Techniques, Security Improvement, Design for Software Security

Example of a complex code analytics
Current Research: Software Engineering as a Service (SEaaS)

- Software Requirements Engineering as a Service - Software security requirements management as an emerging cloud computing service (IJIM paper April 2016)

- Software Project Management as a Service (Collaboration with National Institute of Technology Karnataka, Surathkal Mangalore - Karnataka, India, http://www.nitk.ac.in/people/k-chandrasekaran)

- Quality of Service (QoS) as a Service

- Software testing as a Service (STaaS)
As we can see, there is a continued skill shortage and a rapid change in software technology is faster than predicted.

Why SE for Service and Cloud Computing?

Remember, boom and burst of dot com in the early 2000 and failures of software systems and cybersecurity attacks (we can’t protect systems by adding security patches we have design for it) suggests lack of adopting systematic approach to developing services.

My Personal Moto Learned from childhood: As Avvaiyar (a Tamil Lady poet from 1st-2nd Century of C.E (roughly 2000 years ago Common Era or A.D) wrote (wikipedia):

"Katrathu Kai Mann Alavu, Kallathathu Ulagalavu"

meaning roughly "What you have learned is only a handful; What you haven't learned is the size of the world"
PART 1: WHAT & WHY SOA?, SERVICE COMPUTING, CLOUD COMPUTING, IOT, BIG DATA, WIRELESS SENSOR NETWORKS
What is SOA (So-Ah)?

- ebizQ [http://www.youtube.com/watch?v=sRFwswaJpD0](http://www.youtube.com/watch?v=sRFwswaJpD0)
- Prof Paul Strassmann [http://www.youtube.com/watch?v=nRayJZmj2oY&feature=related](http://www.youtube.com/watch?v=nRayJZmj2oY&feature=related)
- Fundamentals of service delivery platforms – excellent foundation to SOA must watch video talk Fundamentals of service delivery platforms [http://www.youtube.com/watch?v=m1z22GFvsYg](http://www.youtube.com/watch?v=m1z22GFvsYg)
- Kent Mitterer [http://www.youtube.com/watch?v=jl5oFtNwJ9Q&feature=related](http://www.youtube.com/watch?v=jl5oFtNwJ9Q&feature=related)
- Why study SOA? Technology transition; Increasingly software is moved to service oriented (see presence of web service and cloud computing)
- 2015 CEFRIEL - Service Centric Systems Engineering (SeCSE)
What is SOA? Is it a New Science and Engineering?

My definition: SOA is a way of architecting and structuring software assets, components, and services using message passing as the core design principle to maximise reuse.

SOA is a logical way of designing a software system to provide services to either end-user applications or to other services distributed in a network, via published and discoverable interfaces.
Common Sense Definition

**FIGURE 1.2**
SOA Consumer and Service

**FIGURE 1.3**
SOA Consumer, Intermediary, and Service
The Future is here!

Why SOA? Multitude of devices, seamless data, intelligence, multitude of software, systems, services, and platform integration, and predictions. The Future is here!

SOA is a formalised way of integrating applications existing traditional applications and legacy systems) into an enterprise architecture and hence suitability for connecting IoEs
YESTERDAY: GADGETS ARE EVERYTHING

Cool toys…

Too bad they can’t talk to each other…
TODAY: COMMUNICATION IS EVERYTHING
Tomorrow: Service is Everything: they communicate, compose new services, and self recover themselves.

MC10’s Biostamp: wireless health reading data

A Robot Penguin chick to monitor others (IoT)
Universal Theme: Seamless Data, intelligent and ubiquitous interactivity is a key theme across all sectors

**Manufacturing**: Intelligent interconnectivity across the enterprise for enhanced control, speed and efficiency (Industrial IoT)

**Retail**: Highly personalized customer experience across channels and devices

**Healthcare**: Integrated and smart patient care systems and processes

**Banking and Finance**: Seamless customer experience across all banking channels

**Automotive**: V2V and V2I communication, Automatic Cars emerging into market much faster than predicted
Wireless Sensor Networks

Natural Flood & Fire Disasters

Application of Service and Cloud Computing
Bacia Hidrográfica do Rio Mondego
Wireless Sensor Network (WSN) for Predicting Natural Disasters

Challenges on Networked WSN and use of Drones for during and recovery phase of natural floods disasters.

- Optimal energy usage of sensors
- Computational Intelligence and Data Analytics – Some predication models to be built-in
- Three types of sensors: Temperature, Rainfall, and Pressure
- Ongoing project with Portugal and other partners
Use of Drones for Air Surveillance, Monitoring and Recovery: Floods, Fire Recovery, Crops Monitoring
The architecture of a typical wireless sensor network

WSN/IoT for Fire Alert

Temp Sensor DS18B20

Arduino Processor with GSM Module and Temperature Sensor DS18B20

http://www.instructables.com/id/Temperature-with-DS18B20/
Service Providers build services and offer them via an intranet or Internet. They register services with service brokers and publish them in distributed registries. Each service has an interface, known as contract and functionality, which is kept separate from the interface. The Service Consumers search for services (based on some criteria) - when found, a dynamic binding is performed. In this case, the service provides the consumer with the contract details and an endpoint address. The consumer then invokes the service.

Services, implemented as Web Services (WS) are delivered using technologies such as eXtensible Markup Language (XML), Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP) and Universal Description Discovery and Integration (UDDI).
Enterprise Service Buses (ESBs) build on MOM (message-oriented middleware) to provide a flexible, scalable, standards-based integration technology for building a loosely coupled, highly-distributed SOA. ESBs contain facilities for reliable messaging, web services, data and message transformation, content-based ‘straight through’ routing. Along with web services, ESBs are proving to be the major technical enablers for actual SOA projects.
Some key challenges are: Enterprise Integration, Service Reuse, Service Design Patterns, Service management, SE for SOA, Metrics, Architecture Driven Services, Design for Service Security
PART 2: SE FOR CLOUD COMPUTING VS CLOUD SE
SE for Cloud vs Cloud SE

- **SE for Cloud Data Centres (DC)** should focus on engineering approaches to design and development of cloud data centres: service development process, methods, developing reusable services, systematic approaches to cloud deployment, management, pricing, design for scalability, elasticity and sustainability that needs to be build-in, tested and deployed by cloud providers.

- **Cloud SE** should focus on engineering approach to developing new services offered by a cloud with emphasis on build-in for scalability, service reusability, and elasticity.
Challenges for Cloud Software Engineering

- **Systematic SE Approach**: Build a software development environment that radically simplifies hosting scientific applications on a range of clouds (Cloud federation & Multi-clouds)

- **Service Reuse**: Build applications that make use of the cloud providers APIs (PaaS) to access common services.

- Challenge is using existing PaaS APIs for computationally-intensive applications
  - Systems are set up to support web-based applications
  - Google Apps – 30 second time limit on tasks
  - Significant application change required

- **Service Reuse & Design for Scalability**: Investigate how to adapt applications that are computation/data intensive to run within the constraints set by the PaaS interfaces from cloud providers

- **Design for Architecture Reuse**: Application architecture designed for the cloud: Underlying infrastructure management done in the program itself, depending on performance and cost requirements. What abstractions do we need for this?

- **Three grand challenges**
  - Programming models for the cloud
  - Building a PaaS for high performance/throughput computing
  - Cloud-aware software development environments

Service-Oriented Software Engineering
Programming models for the cloud: yet another cloud programming language?

- Are there programming models that can take advantage of elasticity and massive parallelism?
- – Invent new models for parallel computation
- – Adapt existing problems to the current map/reduce programming model
- How do we adopt efficient cloud resource management algorithms?
- What about energy efficiency of the clouds?
- How do we cater for multi-tenancy and pricing models?
Cloud Computing

Introduction to Cloud

https://app.box.com/s/h0thc07ulmpr4g4s12penhg9f6us44td
Cloud Characteristics

- Grid infrastructure
- Pay-per-service
- Open standard
- Security
- Resource management
- Services & SLA
- Virtualisation
- Data Integrity
- Service customisation
- Scalability
- Resilient
- Performance
- Availability
- Elasticity

Performance

Availability

Scalability

Resilient

Service customisation

Data Integrity

Virtualisation

Cloud

Resource management

Services & SLA

Security

Open standard

Pay-per-service

Grid infrastructure
Cloud Services

Cloud Clients
- Web browser, mobile app, thin client, terminal emulator, ...

SaaS
- CRM, Email, virtual desktop, communication, games, ...

PaaS
- Execution runtime, database, web server, development tools, ...

IaaS
- Virtual machines, servers, storage, load balancers, network, ...

Cost Efficiency

Control
Exercise 1: Draw a feature model for Amazon AWS services: EC2 and S3 services http://aws.amazon.com/

Key software development activities—on-premise versus cloud
SOA Development Structure
Cloud Project Management: Service Cost Estimation and Cloud Economics
Economics for traditional SDLC

Incorporating changes at a later stage of SDLC increases the cost of the project exponentially.

Adding more number of programmers at a later stage does not solve the schedule problem as increased coordination requirement slows down the project further. It is very important that requirements gathering, planning, and design of the software are done involving all the parties from the beginning.
Impact of Cloud Computing on Software Engineering

- Existing software process models and framework activities are not going to be adequate unless interaction with cloud providers is included.

- The cloud providers can help in answering these questions about
  - (1) how many developers are needed, (2) component reuse,
  - (3) cost estimation, (4) schedule estimation, (5) risk management, (6) configuration management, (7) change management, and (8) quality assurance (QoS)

- Instead of KOLC/FP (Function points) – service points metrics?
Especially when applications are deployed as “software as a service” or “SaaS” model, they may have occasional workload surge not anticipated in advance.
Service Cost Estimation Techniques
A Service is a unit of solution logic to which service-orientation has been applied to a meaningful extent. It is a container for a collection of related functions.

\[
\text{Service Point (SP)} = \sum_{i=1}^{n} (P_i \times P)
\]

Where \( P_i \) is an infrastructure factor with empirical value that is related to the supporting infrastructure, technology and governance processes. \( P \) represents a single specific service's estimated size that varies with different service types, including existing service, service built from existing resources, and service built from scratch.

Other approaches

Cost of SOA = (Cost of Data Complexity + Cost of Process Complexity + Cost of Service Complexity + Enabling Technology Solution)
Service Complexity vs. Project Complexity

- Service Complexity
  - Functional Complexity
  - QoS Complexity
  - Service Operation Complexity

- Project Delivery Complexity
  - Service Interface Complexity
  - Environment Complexity
  - Service Interface Development Complexity

Functionality
- Request Message Size
- Response Message Size
- Data Translation Complexity
- Business Logic Complexity
- Data Access Complexity
- Domain Objects/Entities Used
- Service Invocation Complexity
- Fault/Error Handling Complexity
- Infrastructure Access Complexity
- Service Participants
  - Integration Actors

Quality of Service
- Interaction Type Objectives
- Response Time Objectives
- Data Load Objectives
- Concurrency Objectives
- Scalability Objectives
- Availability Objectives
- Security Objectives
- Reusability Objectives
- Testability Objectives
- Service Monitoring Objective

Environment / Context
- Business Domain Knowledge
- Technology Domain Knowledge
- Implementation Paradigm Knowledge
- Requirements Stability
- Team Dynamics
- Service Governance Controls
- Tool Support for Build and Deployment
Cost estimation for cloud services

### Table 1.2 COCOMO [29]

<table>
<thead>
<tr>
<th>Software proj.</th>
<th>$a$</th>
<th>$b$</th>
<th>$c$</th>
<th>$d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organic</td>
<td>2.4</td>
<td>1.05</td>
<td>2.5</td>
<td>.38</td>
</tr>
<tr>
<td>Semidetached</td>
<td>3.0</td>
<td>1.12</td>
<td>2.5</td>
<td>.35</td>
</tr>
<tr>
<td>Embedded</td>
<td>3.6</td>
<td>1.2</td>
<td>2.5</td>
<td>.32</td>
</tr>
<tr>
<td>Cloud comp.</td>
<td>4</td>
<td>1.2</td>
<td>2.5</td>
<td>.3</td>
</tr>
</tbody>
</table>

Effort Applied = $a$(KLOC)$^b$ [man months]

Development Time = $c$(Effort Applied)$^d$ [months]

No. of People = Effort Applied / Development Time [no.]
Extended COCOMO for SW effort estimation
Extended COCOMO for SW dev. time
Service Oriented Development Process

- Business Process Identification, Modelling, Simulations
- Service Requirements Engineering & Task Modelling
- Service Design & Architecture (including CBSE)
- Service Implementation, Testing, & Deployment

Diagram:
- Collect reqmts
- Model process
- Design architecture
- Develop services
- Assemble process
- Deploy to WPS
- Test
- QA engineer
- Integration developer
- Java developer
- Architect
- Business Analyst
- Monitor process

Tools:
- ReqPro
- BPEL/WSDL
- UML
- RSA
- EJBs
- RAD
- WID
- WPS
- Testing
- R. testing suite
SOA Lifecycle

Service Lifecycle and SOA Roles
Cloud service security development process with build in security – Our Systematic Approach to adopt BSI as part of CCAF

- Requirements Engineering for cloud applications (services & security)
  - Cloud service security requirements
  - Cloud business process security vulnerabilities

- Conduct BPM
  - SLAs security rules and risks

- Identify SLAs
  - Cloud service security specific components & interfaces

- Design service components
  - Cloud service security testing

- Test & Deploy

Build Cloud Security In – Cloud service development with build-in security
BPMN

• Initial process models: Actors/roles/Workflows
• Detailed workflows
• Service Task modelling
• UI prototyping
• Process Simulation:
  • Configure Resources need for tasks
  • Load profiles in sec/min/days/no.of instances
  • Start the Process Simulation as a Service (PSSaaS)

SOA Requirements with use case modelling, story cards, (Agile), Story Boards, CRC Cards, Feature-Oreinted modelling

SOA Design with Service Component Models

SOA Implementation with SOAP/RESTful

SOA Test & Deliver
As an Architect, you will need to categorise services therefore you will be able to place them in the appropriate architecture layers on the right.
Orthographic Software Modelling (OSM)

The figure shows three distinct views of a software system being projected along three dimensions, to highlight the analogy with the CAD of physical artifacts. However, in the software space, there is no restriction on the number of dimensions that can be defined and the number of choices that can be made in each dimension. This is based on the multi-dimensional modelling work of korbA.
Invitation Call for Chapters on Exploring Security in Software Architecture and Design, IGI Global

The proposed book is very essential for establishing secure software design as it is vulnerable to 50% security flaws (refer IEEE secure design guide), https://www.synopsys.com/content/dam/synopsys/sig-assets/whitepapers/top-10-software-security-design-flaws.pdf. Consider following topics but not limited to:

- Best practice security design guidelines
- BSI (build in security design flaws)
- Design for Security
- Architectural security analysis
- Architecture security risk analysis (ARA)
- Security testing of software and systems architecture,
- Architecture security evaluation, validation, and verification, etc.
- SOA security
- Enterprise Architecture Security
- Cloud Architecture Security

The book is published by IGI Global and submit your chapter by 15th September 2017.

AUTHOR’S GUIDE TO PREPARING CHAPTERS (http://www.igi-global.com/publish/contributor-resources/before-you-write/

AUTHOR’S CHAPTER SUBMISSION CHECKLIST (http://www.igi-global.com/publish/contributor-resources/edited-book-author-checklist/

Please submit your enhanced article in Microsoft Word Format with the word “Enhanced” in the file name through the eEditorial Discovery® system by no later than 15th September 2017. Please use the following link when submitting your enhanced article:

https://www.igi-global.com/submission/?returnurl=%2fsubmission%2fbooks%2f

Research collaborations will enhance our community of EA Research Activities.
The SOA success factors can be summarized as follows:

- Service computing is the future
- Systematic approach to developing services can avoid disasters
- We can learn from mistakes of the past
References/Reading List

- Erl, T (2005) SOA: concepts, technology, design, Prentice Hall/Pearson Education