university of groningen

Cloud-based Applications Engineering Principles

Vasilios Andrikopoulos

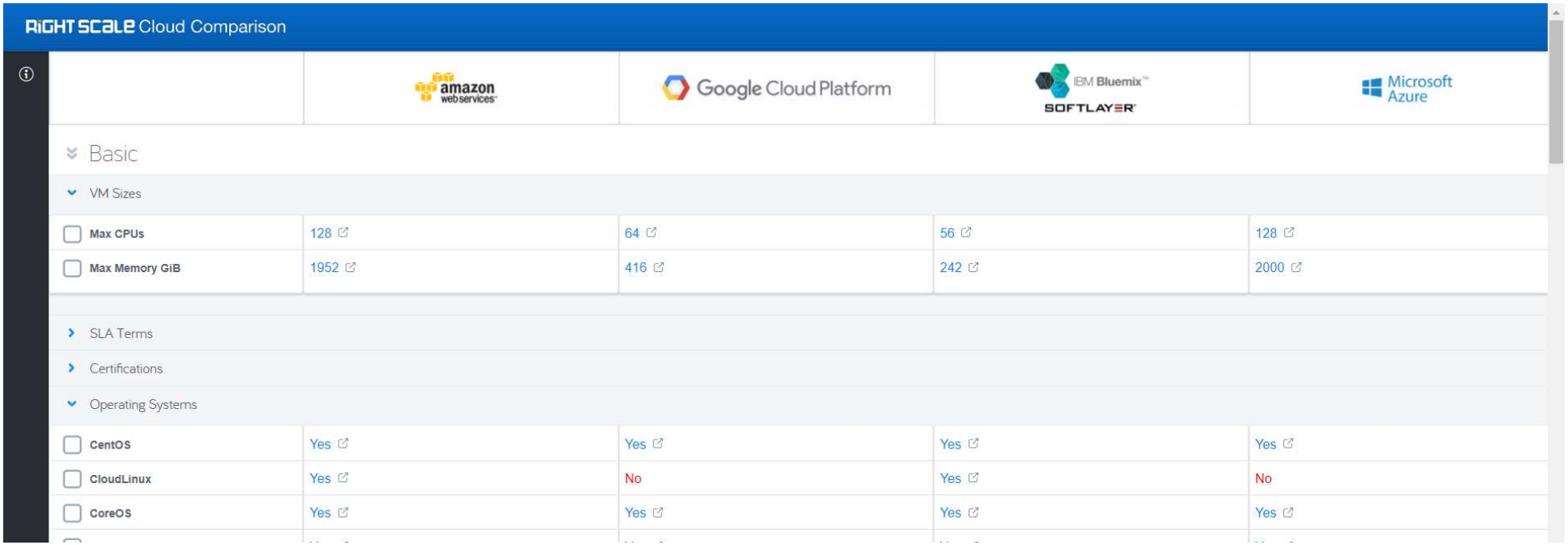
VSS 2017

On the relation of Software Architecture and DevOps/Continuous Delivery

university of groningen

Context – the state of the Cloud

- Cloud as the *de facto* platform for software
- Early confusion resolved by NIST SP 800-145
- Multiple (similar) offerings by the #Stacks



Related movements

- **Movement #1:** DevOps
  - CD/CI frameworks + deployment automation tools = shortened dev cycle + agile practices
  - OS-level virtualization a la Docker -> applications as independent software stacks

- **Movement #2:** Microservices
  - Loosely coupled component lifecycles

university of groningen

Key message

- *Software development in practice has changed, software engineering (research) should do the same*

- Scoping: Cloud-based applications, i.e. both cloud-enabled and cloud-native

Def: Cloud-based applications (CBAs) are applications that **rely on** one or more **cloud services** in order to be able **to deliver their functionality** to their users

university of groningen

Challenges for CBA Engineering

Amazon Relational Database Service English

API Reference (API Version 2014-10-31)

AWS Documentation » Amazon Relational Database Service (RDS) Documentation » API Documentation » Actions » CreateDBInstance

View PDF | Go to the forums

CreateDBInstance

Creates a new DB instance.

Request Parameters

For information about the common parameters that all actions use, see Common Parameters.

AllocatedStorage

The amount of storage (in gigabytes) to be initially allocated for the database instance.

Type: Integer

MySQL

Constraints: Must be an integer from 5 to 3072.

PostgreSQL

Constraints: Must be an integer from 5 to 3072.

Oracle

Constraints: Must be an integer from 10 to 3072.

SQL Server

Constraints: Must be an integer from 200 to 1024 (Standard Edition and Enterprise Edition) or from 20 to 1024 (Express Edition and Web Edition)

Type: Integer

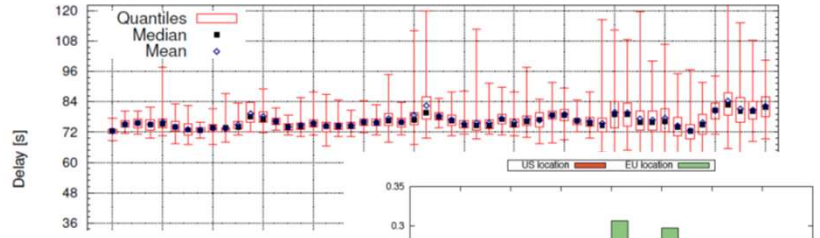
Required: No

AutoMinorVersionUpgrade

Indicates that minor engine upgrades will be applied automatically to the DB instance during the maintenance window.

Default: true

Type: Boolean



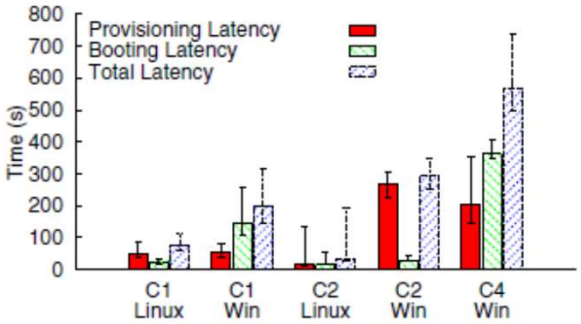
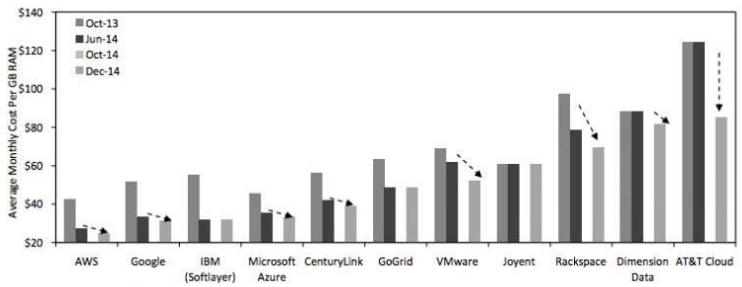


Exhibit 14: Average Monthly Cost / GB RAM across various RBC Use Cases (excluding support costs)



(excluding optional support costs)
Average Monthly Cost Per GB RAM across various RBC Use Cases

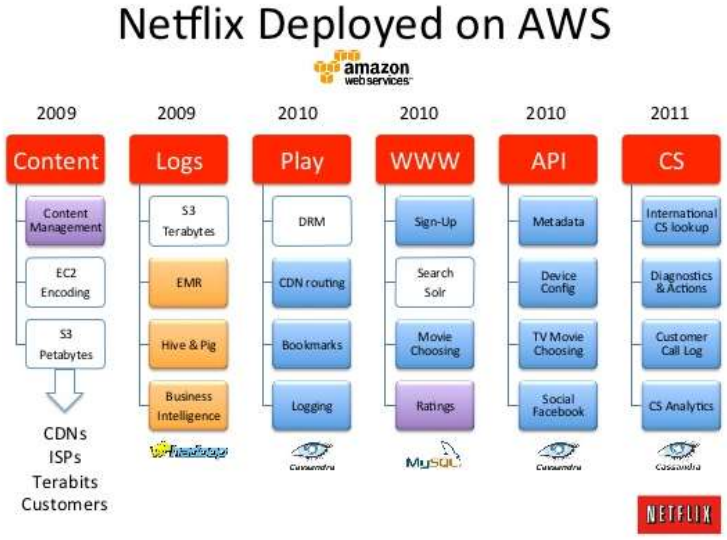
	Oct-13	Jun-14	Dec-14	Change
AWS	$42	$27	$25	-8%
Google	$52	$34	$32	-6%
IBM (Softlayer)	$55	$32	$32	0%
Microsoft Azure	$46	$35	$34	-5%
CenturyLink	$56	$42	$39	-7%
GoGrid	$63	$49	$49	0%
VMware	$69	$62	$52	-15%
Joyent	$61	$61	$61	0%
Rackspace	$98	$79	$70	-11%
Dimension Data	$88	$88	$82	-8%
AT&T Cloud	$125	$125	$85	-32%

* Change Dec-14 vs. Jun-14

(including high-touch support)
Average Monthly Cost Per GB RAM across various RBC Use Cases

	Oct-13	Jun-14	Dec-14	Change
AWS	$53	$38	$35	-6%
Google	$93	$69	$68	-2%
IBM (Softlayer)	$56	$32	$32	0%
Microsoft Azure	$70	$60	$58	-3%
CenturyLink	$56	$46	$43	-7%
GoGrid	$75	$60	$60	0%
VMware	$79	$71	$61	-13%
Joyent	$72	$72	$72	0%
Rackspace	$108	$89	$70	-22%
Dimension Data	$88	$88	$81	-8%
AT&T Cloud	$129	$129	$88	-32%

Source: RBC Capital Markets, Company Reports



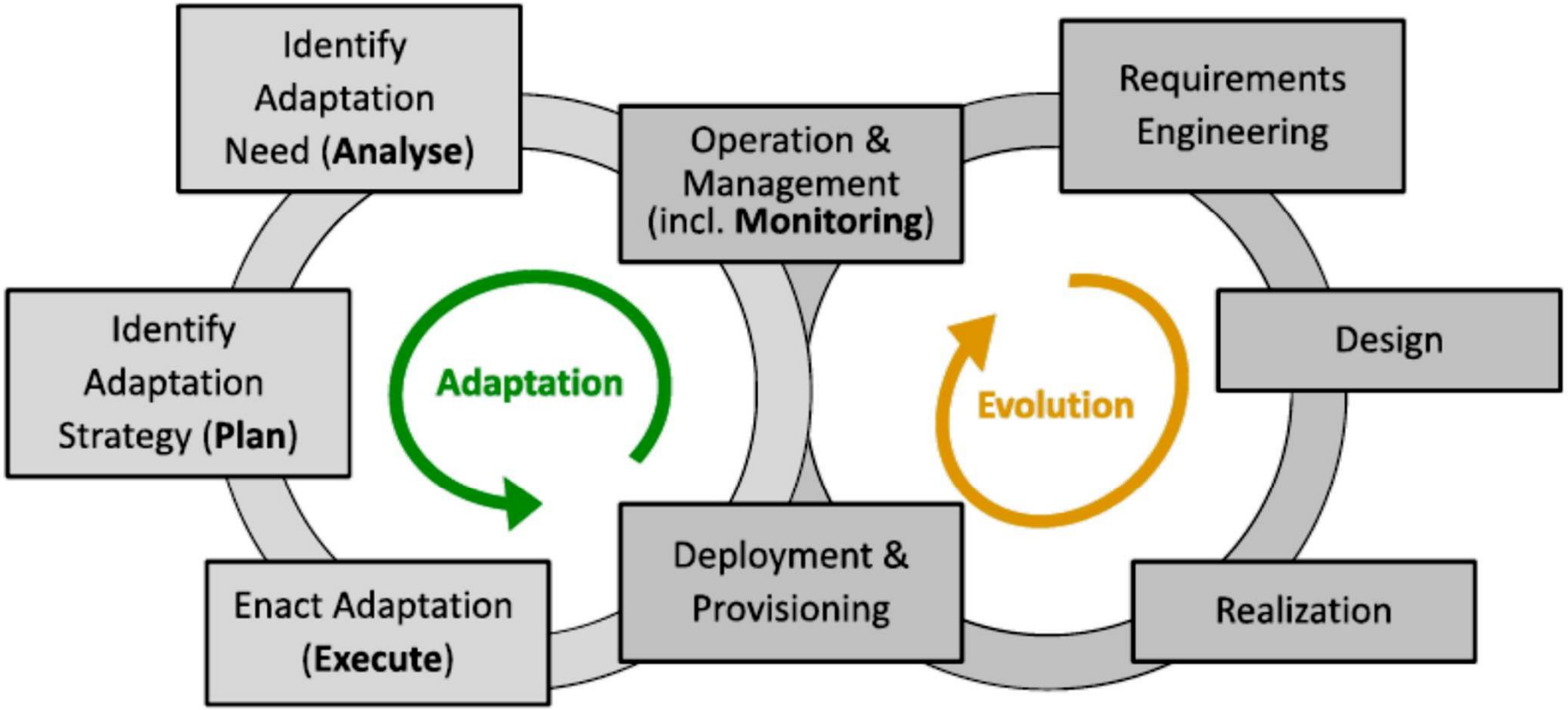
university of groningen

Requirements on the solution space

1. CBA engineering should incorporate *service-orientation concepts* like composition
2. System design should be based on *evolving dynamic topologies*
3. *Self-* characteristics* are essential in dealing with provider-induced variability
4. *Awareness of consumed resources* should be enabled for both development and operation

university of groningen

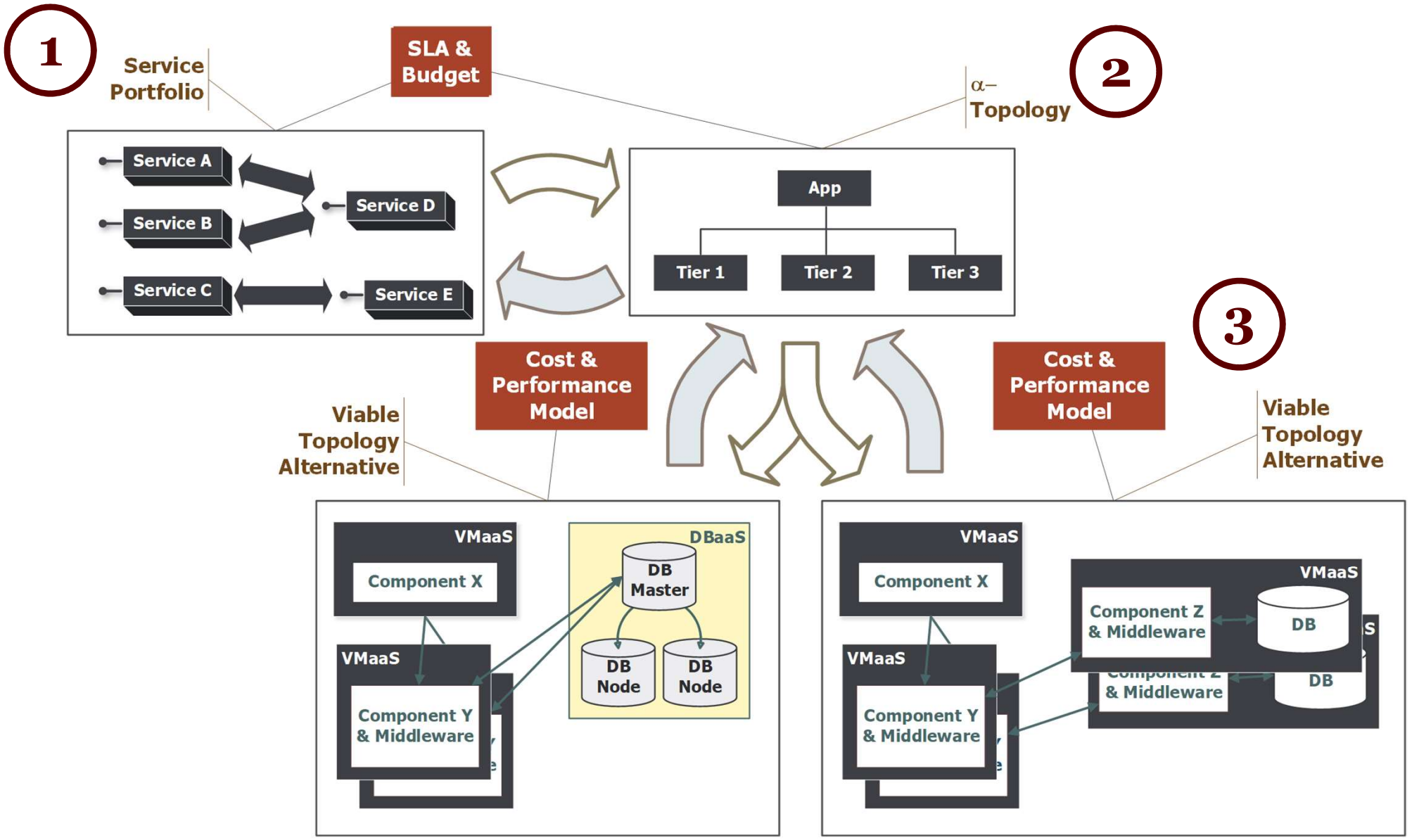
The S-Cube SBA Reference Lifecycle



S-CUBE http://s-cube-network.eu/

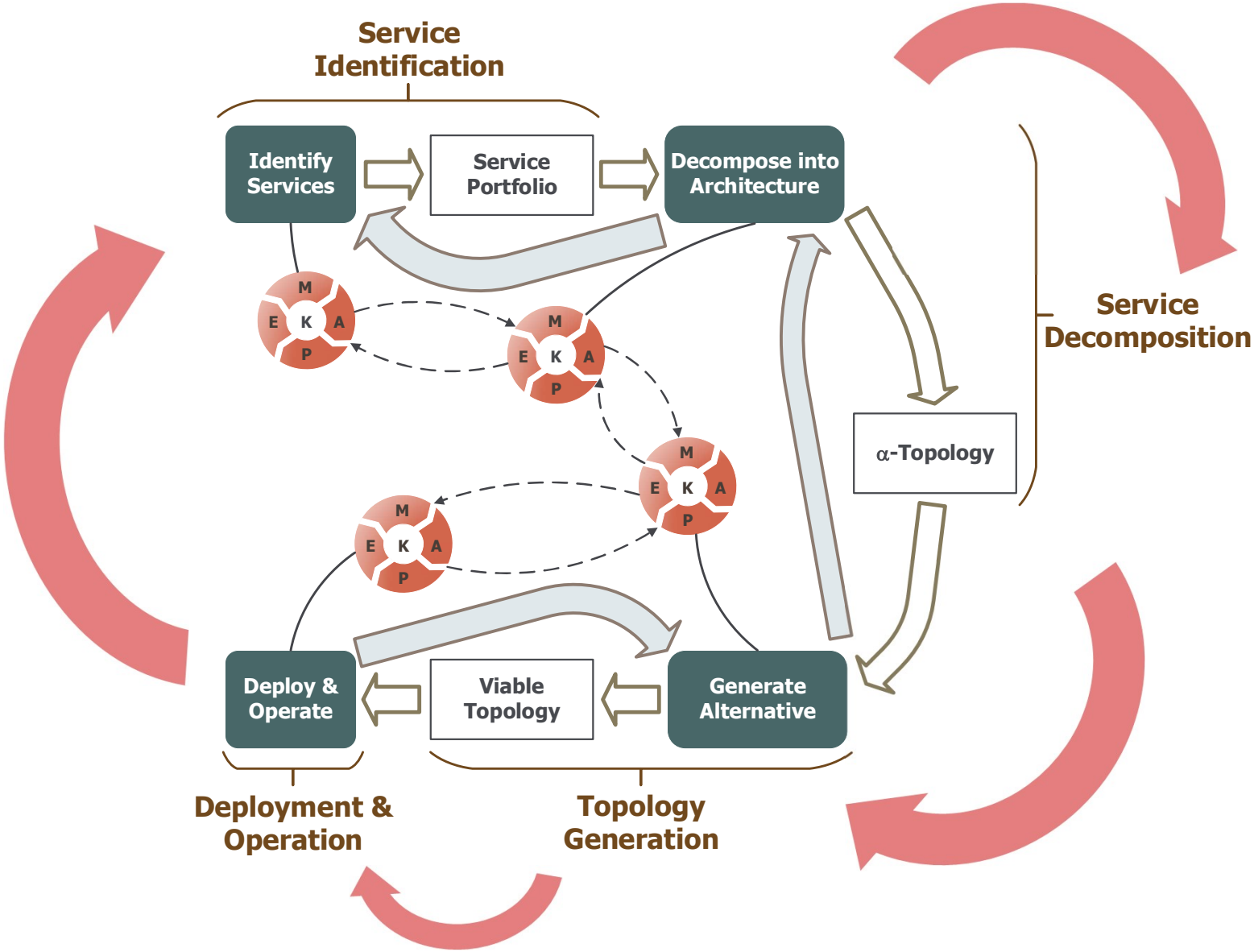
university of groningen

The proposed CBA lifecycle



university of groningen

Phases of the CBA lifecycle in a nutshell



university of groningen

Principles of CBA engineering (proposal)

#1: Architectural decisions are informed by cost at scale

#2: Transitioning between viable topologies should be fast and easy

#3: No need for separation between design and run time anymore

#4: Optimizing for a noisy environment is sub-optimal and potentially unnecessary

university of groningen

Principles of CBA engineering (proposal)

#1: Architectural decisions are informed by cost at scale

#2: Transitioning between viable topologies should be fast and easy

#3: No need for separation between design and run time anymore

#4: Optimizing for a noisy environment is sub-optimal and potentially unnecessary

university of groningen

Principles of CBA engineering (proposal)

#1: Architectural decisions are informed by cost at scale

#2: Transitioning between viable topologies should be fast and easy

#3: No need for separation between design and run time anymore

#4: Optimizing for a noisy environment is sub-optimal and potentially unnecessary

university of groningen

Principles of CBA engineering (proposal)

#1: Architectural decisions are informed by cost at scale

#2: Transitioning between viable topologies should be fast and easy

#3: No need for separation between design and run time anymore

#4: Optimizing for a noisy environment is sub-optimal and potentially unnecessary

university of groningen

Principles of CBA engineering (proposal)

#1: Architectural decisions are informed by cost at scale

#2: Transitioning between viable topologies should be fast and easy

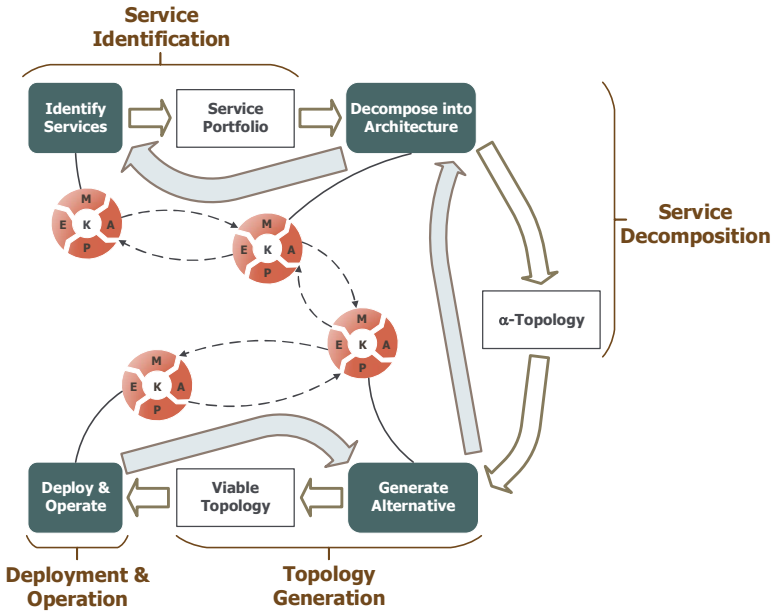
#3: No need for separation between design and run time anymore

#4: Optimizing for a noisy environment is sub-optimal and potentially unnecessary

university of groningen

Conclusions

- Adoption of cloud computing + DevOps (incl. virtualization) + microservices → need for new take on CBA engineering
- CBA lifecycle as interconnected loops
- Open issues
  - Security
  - QA
- Future work
  - Tooling
  - Validation



Reach me at:

v.andrikopoulos@rug.nl
https://vandriko.github.io
@v_andrikopoulos