Designing a business platform using Microservices

Rajendra Kharbuja, Master Thesis - Initial Presentation

Software Engineering für betriebliche Informationssysteme (sebis)
Fakultät für Informatik
Technische Universität München

wwwmatthes.in.tum.de

Prof. Dr. Florian Matthes
[Supervisor]
Manoj Mahabaleshwar
[Advisor]

Andrea Stubbe
[Advisor]

Definition

- application deployed as a single artifact irrespective of internal structure

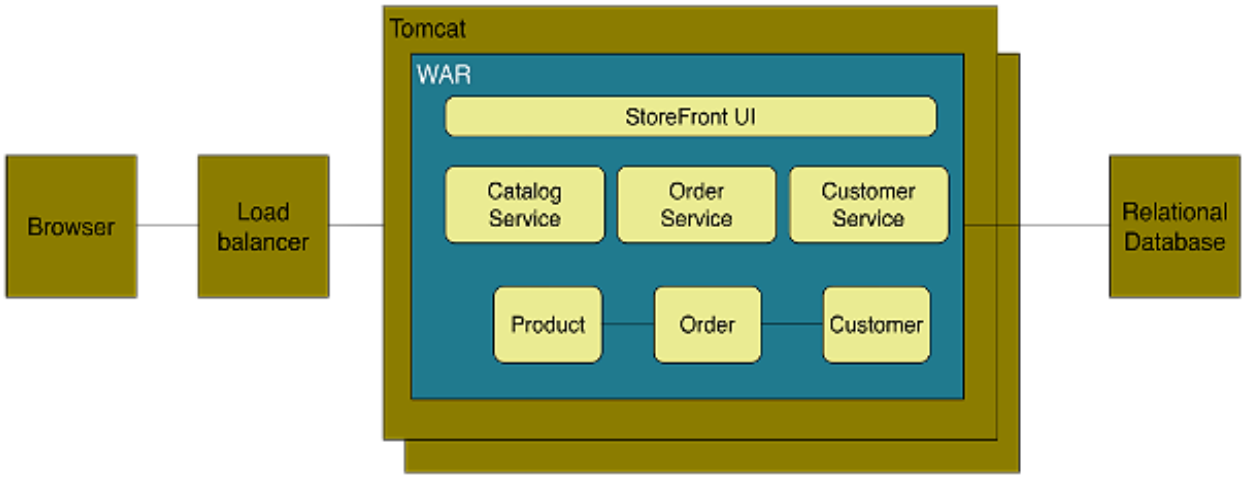


Figure: Online Store Example of Monolith Architecture

Monolith Architecture Style

sebis

Advantages

- Easy to develop and test
- Deployment is easy
- Scaling is simple

Disadvantages

- **Limited Agility due to difficult continuous delivery**
  - Single codebase, deployment of small change needs whole application to be deployed
  - Affects continuous delivery, especially when multiple deployment per day
  - May lack of clear modular boundary and changes get relayed
- **Decrease in Productivity due to understandability**
  - Understanding of application is difficult due to overwhelming volume of codebase, esp for new developer
- **Long-term Commitment to Technology Stack**
  - **Technology chosen during analysis phase**
- **Limited Scalability due to unavailable option for scaling of individual units**
  - Only one option ie, Horizontal Scaling
  - Independent scaling of components not possible

Definition

"decompose an application functionally into a set of **collaborating** services, each with a set of narrow, **related functions**, developed and deployed independently, with its **own database**."

[C. Richardson. 2014]

"a style of software architecture that involves delivering systems as a set of very **small**, **granular**, **independent collaborating** services."

[B. Wootton, 2014]

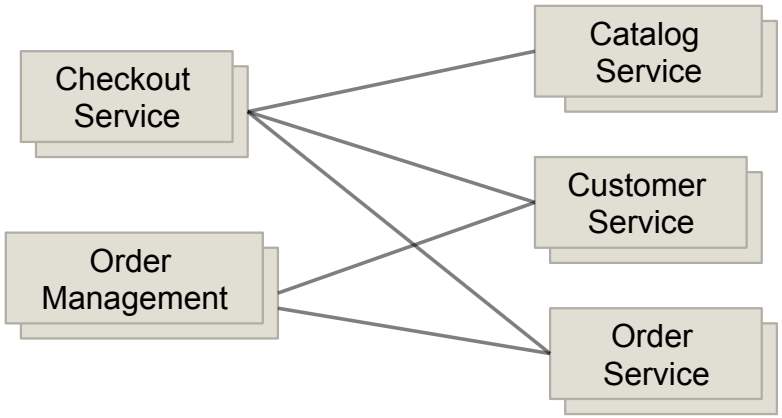


Figure: Functional Decomompostion into microservices

"**Loosely coupled** service oriented architecture with **bounded contexts**."

[A. Cockcroft, 2015]

sebis

Definition

"Microservices are **SOA** done right"

[M. Fowler and J. Lewis, 2014]

[A. Cockcroft, 2015]

[G. Radchenko, O. Taipale, and D. Savchenko, 2015]

"is an approach to developing **a single application** as a **suite** of **small services**, each **running** in its **own process** and **communicating** with **lightweight mechanisms**, often an HTTP resource API. These services are built around **business capabilities** and **independently deployable** by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in **different programming languages** and use **different data storage technologies.**"

[M. Fowler and J. Lewis, 2014]

Research Questions

sebis

1. What are the parameters that defines the size of a microservice?

2. What are the good practices for defining a microservice architecture?

1. What is the process of creating a microservice architecture?
2. What are the quality attributes of a microservice?
3. How to implement a microservice?
4. How to deploy a microservice?
5. How to maintain a microservice?

Research Process
Phase 1

Data Collection Phase

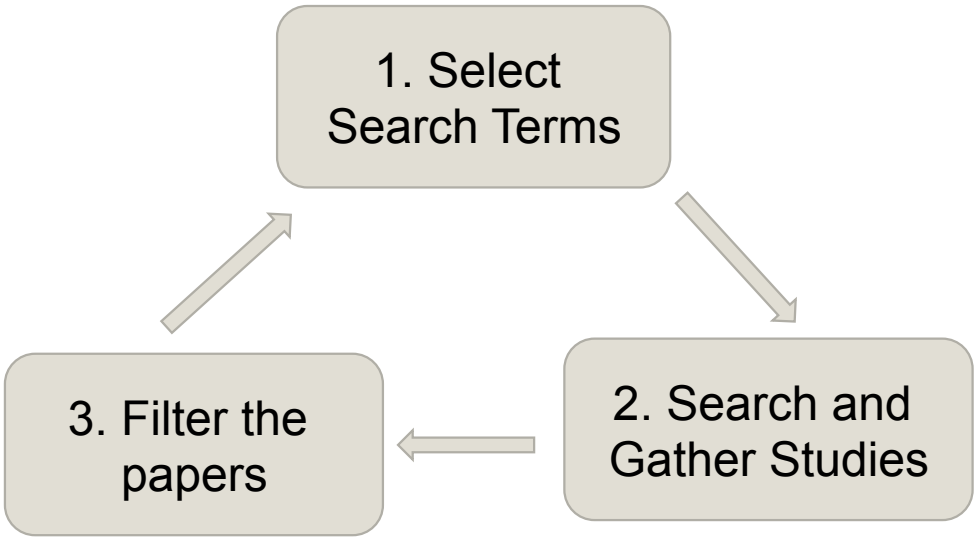


Figure: Data Collection Phase

1. Search Term Selection Strategy

- keywords from research questions and definition
- synonyms of keywords
- accepted terms from academics and industry
- reference discovered from papers selected

2. Resources Used

- google scholar (scholar.google.de)
- IEEExplore
- ACM Digital Library
- Researchgate
- Books
- Technical Articles

Materials Found : 123

3. Study Filteration Criteria

- Is the study relevant to answer the research question?
- Does the study have good base in terms of source as well as references of the past studies?
- Are there any case studies provided to verify the research result derived in the study?

Materials Selected: 64

Keywords And Research Questions

Topics Keywords	Size	Quality of good Microservice	Communication	Process
Collaborating services			✓	
Communicating with lightweight mechanism HTTP			✓	
Loosely coupled, related functions		✓	✓	
Developed and deployed independently				✓
Own database		✓		✓
Different database technologies				✓
Service Oriented Architecture		✓		✓
Bounded Context	✓	✓		✓
Build around business capabilities	✓	✓		✓
Different programming languages				✓

Research Process
Phase 2

sebis

Data Synthesis Phase

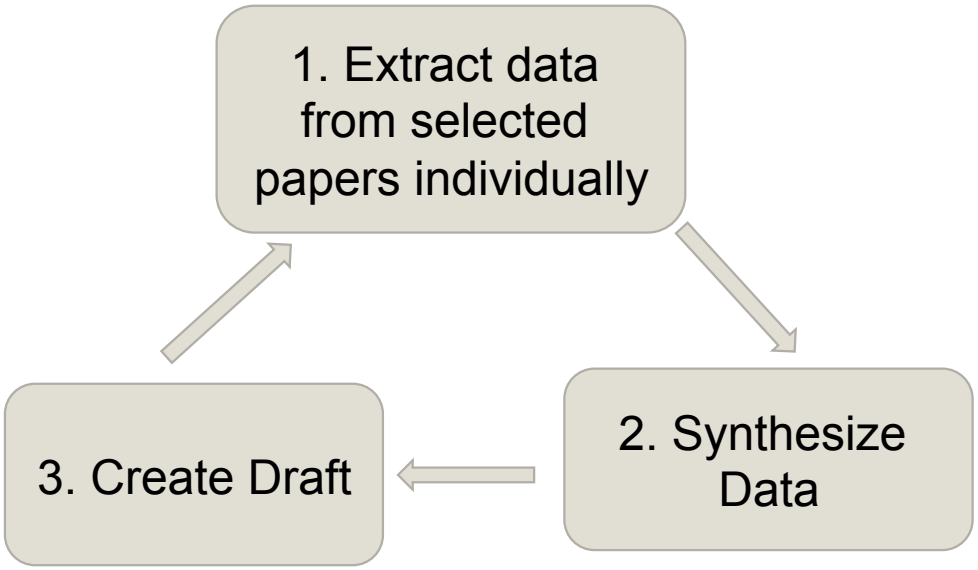


Figure: Data Synthesis Phase

2. Data Synthesis Strategy

- Comparision of the data achieved from papers
- Similaritiies and differences studied

Topic	Found	Selected
Granularity	17	9
Service Attributes	26	10
Service Identification	22	16
Microservices	32	19
Domain Modeling	26	10

Size of a Microservice

Articles Found: 9
Selected: 3

$$ODG = fraction\_of\_total\_weight\_of\_input\_and\_output\_parameters = \left(\frac{\sum_{i=1}^{n_i o} W_{pi}}{\sum_{i=1}^{n_i s} W_{pi}} + \frac{\sum_{j=1}^{n_j o} W_{pj}}{\sum_{j=1}^{n_j s} W_{pj}}\right)$$

$$OFG = complexity\_weightage\_of\_operation = \frac{W_i(o)}{\sum_{i=1}^{|O(S)|} W_i(o)}$$

$$granularity = sum\_of\_product\_of\_data\_and\_functionality\_granularity = \sum_{i=1}^{|O(S)|} ODG(i) * OFG(i)$$

- **ODG is Operation Data Granularity**
- **OFG is Operation Functionality Granularity**

$$SCG = number\_of\_operations = |O(s)|$$

$$SDG = size\_of\_messages = |M(s)|$$

- SCG is Service Capability Granularity
- SDG is Service Data Granularity

$$granularity = \frac{number\_of\_operations}{size\_of\_messages} = \frac{|O(s)|^2}{|M(s)|^2}$$

Granularity of a service is determined by the number of operations, type of parameters of operations and size of messages.

Correct Size of a Microservice

sebis

Single Responsibility Principle

- Microservices should be small and focused on doing one thing well [S.Newman and M. Stine]
- “Gather together those things that change for the same reason, and separate those things that change for different reasons.” [16]
- Small enough to affect only small number of consumers
- Single Responsibility Principle closely related to Coupling and Cohesion. [18, 19]

Autonomy [8, 20, 21, 22]

- “The logic governed by a service resides within an explicit boundary” [24]
- “ The service has control within this boundary and does not depend on other services to execute its governance. It also frees the service from ties that could inhibit its deployment and evolution” [23]
- Big enough to have control upon its business entities

Technology Available

The right value of granularity for an organization is highly influenced by its IT infrastructure. The organization should be capable of handling the complexities such as communication, runtime, infrastructure etc if they choose low granularity. [25, 26]

Quality Attributes for Services

sebis

#	References/ Attributes	[13]	[27]	[15]	[14]	[28]	[29]
1	Coupling	✔	✔	✔	✔	✔	✔
2	Cohesion	✔	✗	✔	✔	✔	✔
3	Autonomy	✔	✗	✗	✗	✗	✗
4	Granularity	✔	✔	✔	✔	✔	✔
5	Reusability	✔	✗	✗	✔	✗	✔
6	Abstraction	✔	✗	✗	✗	✗	✗
7	Complexity	✗	✗	✗	✔	✗	✗

Coupling

$$Coupling = \frac{dependency\_on\_business\_entities\_of\_other\_services}{number\_of\_operations\_and\_dependencies}$$

$$= \frac{\sum_{i \epsilon D_s} \sum_{k \epsilon O(s)} CCO(i,k)}{|O(s)|.K}$$

where,

- $D_s = \{D_1, D_2, ... D_k\}$ is set of dependencies the service has on other services
- $K = |D_s|$
- CCO is Conceptual Coupling betweeen service operations and obtained from BE X EBP (CRUD) matrix table constructed with business entities and business operations, where BE is business entity and EBP any logical process defined in an operation.

$$coupling = \frac{number\_of\_invocation}{number\_of\_services}$$

$$= \frac{\sum_{i=1}^{|O(s)|} (S_{i,sync} + S_{i,async})}{|S|}$$

where,

- $S_{i,sync}$ is the synchronous invocation in the operation O_i
- $S_{i,async}$ is the asynchronous invocation in the operation O_i

$$SOCI(s) = number\_of\_operations\_invoked$$
$$= |\{o_i \epsilon s_i : \exists_{o \epsilon s} calls(o, o_i) \wedge s \neq s_i\}|$$

$$ISCI(s) = number\_of\_services\_invoked$$
$$= |\{s_i : \exists_{o \epsilon s}, \exists_{o_i \epsilon s_i} calls(o, o_i) \wedge s \neq s_i\}|$$

$$SMCI(s) = size\_of\_message\_required\_from\_other\_services$$
$$= |\cup M(o_i) : (o_i \epsilon s_i) \vee (\exists_{o \epsilon s}, \exists_{o_i \epsilon s_i} calls(o, o_i) \wedge s \neq s_i)|$$

where,

- SOCI is Service Operation Coupling Index
- ISCI is Inter Service Coupling Index
- SMCI is Service Message Coupling Index
- $call(o, o_i)$ represents the call made from service 'o' to service 'o_i'

$$coupling(s) = \frac{number\_of\_services\_connected}{number\_of\_services}$$
$$= \frac{n_c + n_p}{n_s}$$

where,

- n_c is number of consumer services
- n_p number of dependent services
- n_s total number of services

Coupling in a service increases with the number of service invocations, number of dependent services and size of messages used.

$$SIDC = \frac{number\_of\_operations\_sharing\_same\_parameter}{total\_number\_of\_parameters\_in\_service} = \frac{n_{op}}{n_{pt}}$$

$$SIUC = \frac{sum\_of\_number\_of\_operations\_used\_by\_each\_consumer}{product\_of\_total\_no\_of\_consumers\_and\_operations} = \frac{n_{oc}}{n_c * |O(s)|}$$

$$SSUC = \frac{sum\_of\_number\_of\_sequentials\_operations\_accessed\_by\_each\_consumer}{product\_of\_total\_no\_of\_consumers\_and\_operations}$$

where,

- SIDC is Service Interface Data Cohesion
- SIUC is Service Interface Usage Cohesion
- SSUC is Service Sequential Usage Cohesion
- n_{op} is the number of operations in the service which share the same parameter types
- n_{pt} is the total number of distinct parameter types in the service
- n_{oc} is the total number of operations in the service used by consumers
- n_{so} is the total number of sequentially accessed operations by the clients of the service

$$SFCI(s) = \frac{number\_of\_operations\_using\_same\_message}{number\_of\_operations}$$

$$= \frac{max(\mu(m))}{|O(s)|}$$

where,

- SFCI is Service Functional Cohesion Index
- the number of operations using a message 'm' is $\mu(m)$ such that $m \epsilon M(s)$ and $|O(s)| > 0$

Cohesion is given by the number of operations sharing same messages, parameters and consumer.

$$complexity = \frac{coupling\_of\_service}{number\_of\_services} = \frac{CS(s_i)}{|S|}$$

- $CS(s_i)$ gives coupling value of a service
- |S| is the number of services to realize the application

$$Complexity(s) = \frac{Service\_Granularity}{number\_of\_services}$$

$$= \frac{\sum_{i=1}^{|O(s)|}(SG(i))^2}{|S|}$$

- |S| is the number of services to realize the application
- SG(i) gives the granularity of ith service

Complexity of the service increases with granularity and coupling.

$$Self-Containment(SLC) = sum\_of\_CRUD\_coefficients\_for\_each\_Business\_entity$$

$$= \frac{1}{h_2 - l_2 + 1} \sum_{i=l_2}^{h_2} \sum_{sr \epsilon SR} (BE_{i,sr} XV_{sr})$$

$$Dependency(DEP) = dependency\_of\_service\_on\_other\_service\_business\_entities$$

$$= \frac{\sum_{j=L1_i}^{h1_i} \sum_{k=1}^{BE} V_{sr_{jk}} - \sum_{j=L1_i}^{h1_i} \sum_{k=L2_i}^{j2_i} V_{sr_{jk}}}{nc}$$

$$\text{autonomy} = \begin{cases} SLC - DEP & \text{if SLC} > \text{DEP} \\ 0 & \text{otherwise} \end{cases}$$

- nc is the number of relations with other services
- $BE_{i,sr} = 1$ if the service performs action sr on the ith BE and sr represents any CRUD operation and V_{sr} is the coefficient depending upon the type of action
- $V_{sr_{jk}}$ is the corresponding value of the action in jkth element of CRUD matrix, it gives the weight of corresponding business capability affecting a business entity
- $l1_i, h1_i, l2_i, h2_i$ are bounding indices in CRUD matrix of ith service

Autonomy of a service increases with degree of control on its business entities and decreases with dependencies.

$$Reusability = number\_of\_existing\_consumers$$
$$= |S_{consumers}|$$

$$Reusability = \frac{Cohesion - granularity + Consumability - coupling}{2}$$

Reusability increases with cohesion and consumers but decreases with granularity and coupling.

Basic Quality Metrics for Services

sebis

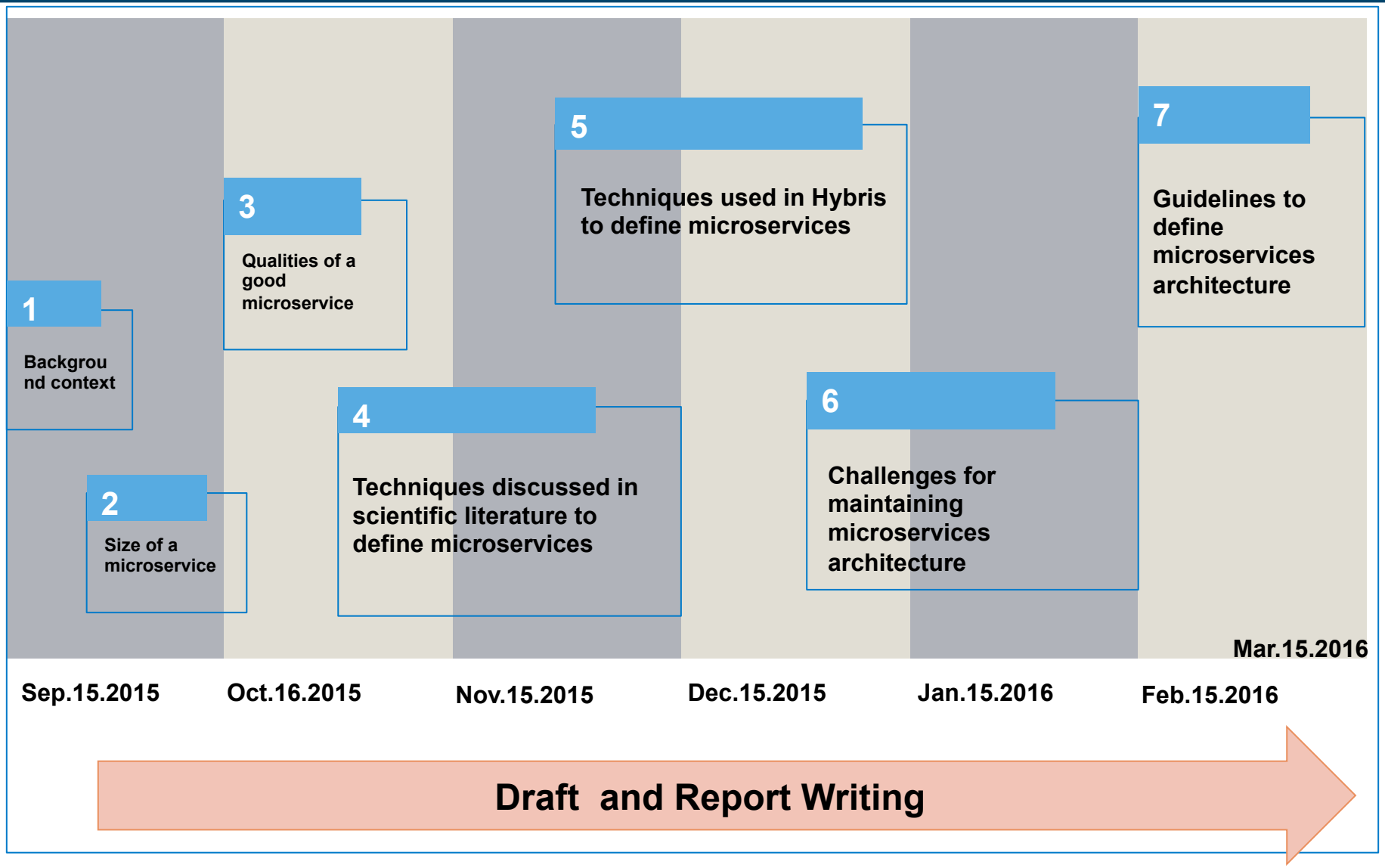
#	Metrics	Coupling	Cohesion	Granularity	Complexity	Autonomy	Reusability
1	number of service operations invoked by the service	+			+	-	-
2	number of operation using similar messages		+				+
3	number of operation used by same consumer		+				+
4	number of operation using similar parameters		+				+
5	number of operation with similar scope or capability		+				+
6	scope of operation			+	+		-
7	number of operations			+	+		-
8	number of parameters in operation			+	+		-
9	type of parameters in operation			+	+		-
10	number and size of messages used by operations	+		+	+	-	-
11	type of messages used by operations		+				+
12	number of consumer services	+			+	-	+
13	number of producer services	+			+	-	-
14	type of operation and business entity invoked by the service	+			+	-	-
15	number of consumers accessing same operation		+				+
16	number of consumer with similar operation usage sequence		+				+
17	dependency degree or imporance of service operation to other service				+		
18	degree of control of operation to its business entities					+	

Current Status and Summary

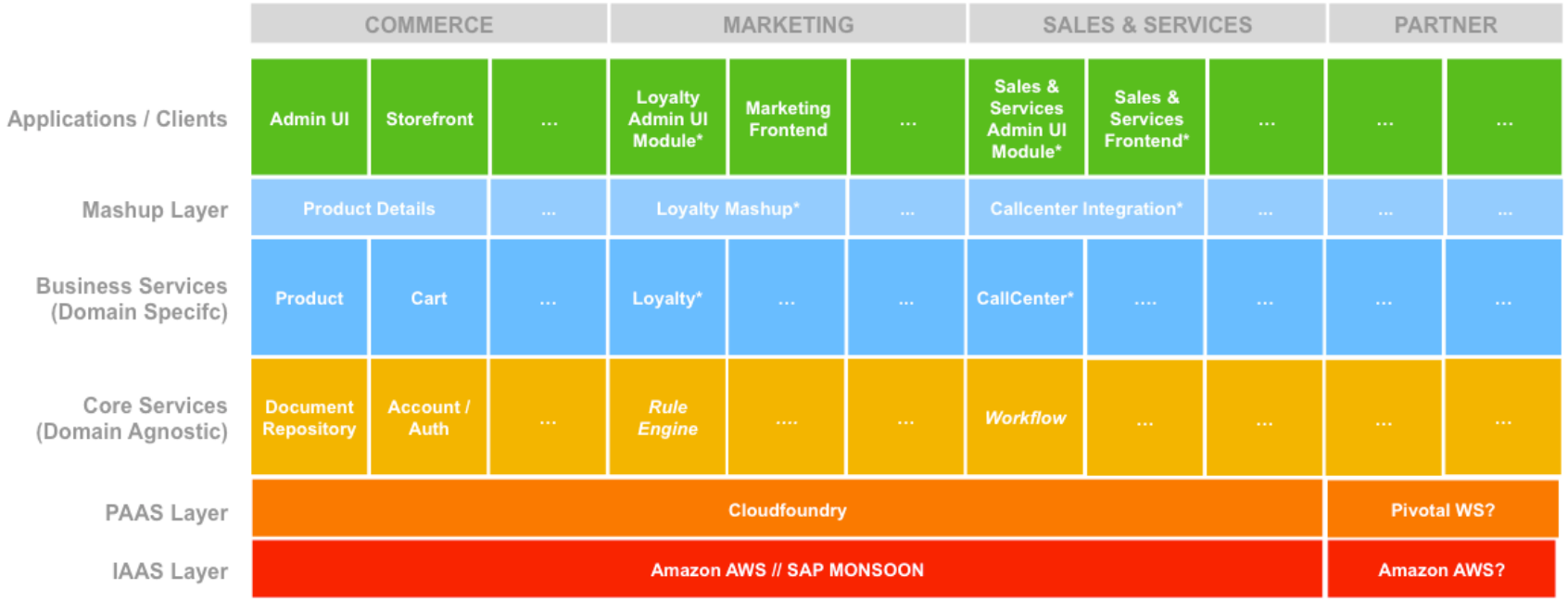
sebis

Research Question	Summary
How can size of a microservice be defined?	• factors determining size • Principles defining a correct size of service listed
What makes any service a good candidate for being a microservice?	• Factors influencing the size defined • attributes to determine a good service • Metrics to calculate the attributes • Interpretation of all metrics from papers into a set of basic metrics • Principles defined for a good service based on quality attributes
What are the techniques defined to create microservice architecture?	• Use case modeling • Domain Driven Design

What next?


1
Backgrou nd context
2
Size of a microservice
3
Qualities of a good microservice
4
Techniques discussed in scientific literature to define microservices
5
Techniques used in Hybris to define microservices
6
Challenges for maintaining microservices architecture
7
Guidelines to define microservices architecture
Mar.15.2016
Sep.15.2015
Oct.16.2015
Nov.15.2015
Dec.15.2015
Jan.15.2016
Feb.15.2016
Draft and Report Writing

Hybris Microservices Architecture



Thank you for your attention!

Any questions?

Bibliography

sebis

1. C. Richardson. Microservices: Decomposing Applications for Deployability and Scalability. May 2014
2. C. Richardson. Pattern: Monolithic Architecture. 2014. url: http://microservices.io/patterns/monolithic.html
3. R. Annett. What is a Monolith? Nov. 2014
4. M. Fowler and J. Lewis. Microservices. Mar. 2014. url: http://martinfowler.com/articles/microservices.html
5. Gupta. Microservices, Monoliths, and NoOps. Mar. 2015
6. S. Abram. Microservices. Oct. 2014. url: http://www.javacodegeeks.com2014/10/microservices.html
7. D. Namiot and M. Sneps-Sneppe. On Micro-services Architecture. Tech. rep.Open Information Technologies Lab, Lomonosov Moscow State University,2014
8. S. Newman. Building Microservices. O'Reilly Media, 2015
9. C. Richardson. Pattern: Microservices Architecture. 2014.
10. B. Wootton. Microservices - Not A Free Lunch! Apr. 2014. url: http://highscalability.com/blog/2014/4/8/microservices- not-a- freelunch.html.
11. A. Cockcroft. State of the Art in Mircroservices. Feb. 2015. url: http://www.slideshare.net/adriancockcroft/microxchg-microservices.
12. G. Radchenko, O. Taipale, and D. Savchenko. Microservices validation: Mjolnirrplatformcase study. Tech. rep. Lappeenranta University of Technology,2015
13. R. Sindhgatta, B. Sengupta, and K. Ponnalagu. Measuring the Quality of Service Oriented Design. Tech. rep. IBM India Research Laboratory
14. B. Shim, S. Choue, S. Kim, and S. Park. A Design Quality Model for Service-Oriented Architecture. Tech. rep. Sogang University, 2008
15. S. Alahmari, E. Zaluska, and D. C. D. Roure. A Metrics Framework for Evaluating SOA Service Granularity. Tech. rep. School of Electronics and Computer Science University Southampton, 2011.
16. http://programmer.97things.oreilly.com/wiki/index.php/The_Single_Responsibility_Principle
17. M. Stine. Jun. 2014. url: http://www.mattstine.com/2014/06/30/microservices-are-solid
18. E. Cobham Brewer url: http://www.objectmentor.com/resources/articles/srp.pdf
19. http://deviq.com/single-responsibility-principle/
20. J. Cramon. Feb. 2014 url: https://www.tigerteam.dk/2014/micro-services-its-not-only-the-size-that-matters-its-also-how-you-use-them-part-1/

Bibliography

21. J. Stenberg. Feb. 2015 url: http://www.infoq.com/news/2015/02/characteristics-microservices-ap
22. A. Rostampour, A. Kazemi, F. Shams, P. Jamshidi, and A. Azizkandi. Measures of Structural Complexity and Service Autonomy. Tech. rep. Shahid Beheshti University GC, 2011.
23. Y. Ma, H.X. Li, and P. Sun, "A Lightweight Agent Fabric for Service Autonomy," Springer, 2007, pp. 63-77.
24. P. Reldin and P. Sundling. Explaining SOA Service Granularity– How IT strategy shapes services. Tech. rep. Linköping University, 2007.
25. P. Herzum and O. Sims. Business Components Factory: A Comprehensive Overview of Component-Based Development for the Enterprise. John Wiley Sons, 2000
26. W. Xiao-jun. Metrics for Evaluating Coupling and Service Granularity in Service Oriented Architecture. Tech. rep. Nanjing University of Posts and Telecommunications
27. Q. Ma, N. Zhou, Y. Zhu, and H. Wang1. Evaluating Service Identification with Design Metrics on Business Process Decomposition. Tech. rep. IBM China Research Laboratory and IBM T.J. Watson Research Center, 2009
28. G. Feuerlicht and J. Lozina. Understanding Service Reusability. Tech. rep. University of Technology, 2007
29. Guidelines for performing Systematic Literature Reviews in Software Engineering.Tech. rep. Keele University, 2007
30. D. Foody. Getting web service granularity right. 2005.
31. A. Goeb and K. Lochmann. A software quality model for SOA. Tech. rep. Technische Universität München and SAP Research, 2011.
32. A. Gupta. Microservices, Monoliths, and NoOps. Mar. 2015.
33. R. Haesen, M. Snoeck, W. Lemahieu, and S. Poelmans. On the Definition of Service Granularity and Its Architectural Impact. Tech. rep. Katholieke Universiteit Leuven.
34. J. K. Lee, S. J. Jung, S. D. Kim, W. H. Jang, and D. H. Ham. Component Identification Method with Coupling and Cohesion. Tech. rep. Soongsil University and Software Quality Evaluation Center, 2001.
35. Q. Ma, N. Zhou, Y. Zhu, and H. Wang1. Evaluating Service Identification with Design Metrics on Business Process Decomposition. Tech. rep. IBM China Research Laboratory and IBM T.J. Watson Research Center, 2009.
36. M. Mancioppi, M. Perepletchikov, C. Ryan, W.-J. van den Heuvel, and M. P. Papazoglou. Towards a Quality Model for Choreography. Tech. rep. European Research Institute in Services Science, Tilburg University.
37. Y.-F. Ma, H. X. Li, and P. Sun. A Lightweight Agent Fabric for Service Autonomy. Tech. rep. IBM China Research Lab and Bei Hang University, 2007.
38. V. D. Naveen Kulkarni. The Role of Service Granularity in A Successful SOA Realization – A Case Study. Tech. rep. SETLabs, Infosys Technologies Ltd, 2008.

Bibliography

39. M. Perepletchikov, C. Ryan, K. Frampton, and Z. Tari. Coupling Metrics for Predicting Maintainability in Service-Oriented Designs. Tech. rep. RMIT University, 2007.
40. M. Perepletchikov, C. Ryan, and K. Frampton. Cohesion Metrics for Predicting Maintainability of Service-Oriented Software. Tech. rep. RMIT University, 2007.
41. C. Richardson. Microservices: Decomposing Applications for Deployability and Scalability. May 2014.
42. B. Shim, S. Choue, S. Kim, and S. Park. A Design Quality Model for Service-Oriented Architecture. Tech. rep. Sogang University, 2008.

CRUD Matrix

sebis

EBP \ BE	customer	Credit	Account receivable note	Order	Discounts	Invoice	Shipping schedule	Draft	Inventory	Warehouse voucher
Add Customer	C	C								
Add an Account receivable note	R	U	C			R				
Check Credit	R	R		R						
Receive order	R			C						
Calculate discounts				R	R					
Check inventory				R					R	
Calculate price				R	R					
Add discounts					C					
Issue invoice	R	R		R		C				
Schedule shipping						R	C			
Issue draft						R	R	C		
Add an Item									C	
Add a warehouse voucher	R					R			U	C

Action	Coefficient
Create(C)	0.4
Read(R)	0.1
Update(U)	0.3
Delete(D)	0.2