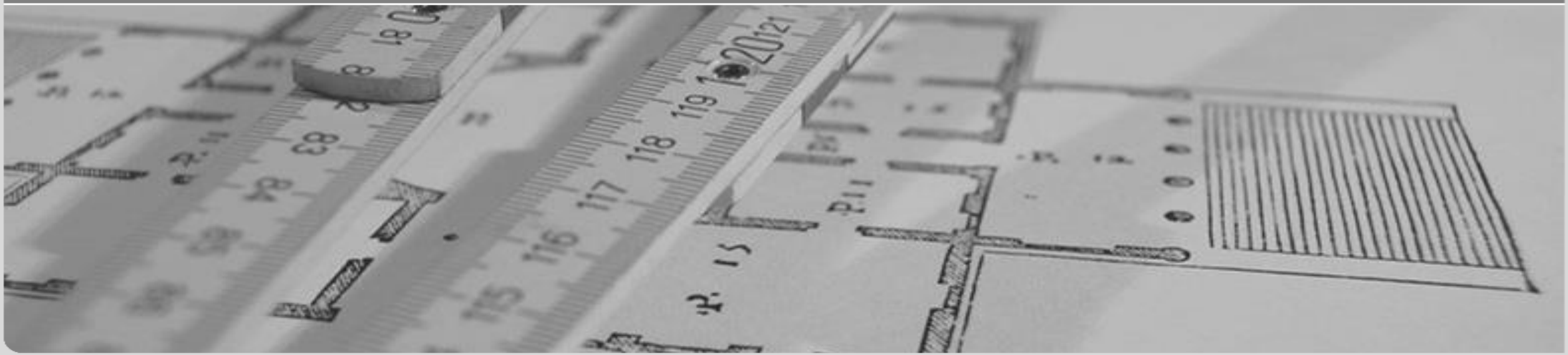
Karlsruhe Institute of Technology

An Approach for Identifying Microservices using Clustering on Control Flow and Data Flow

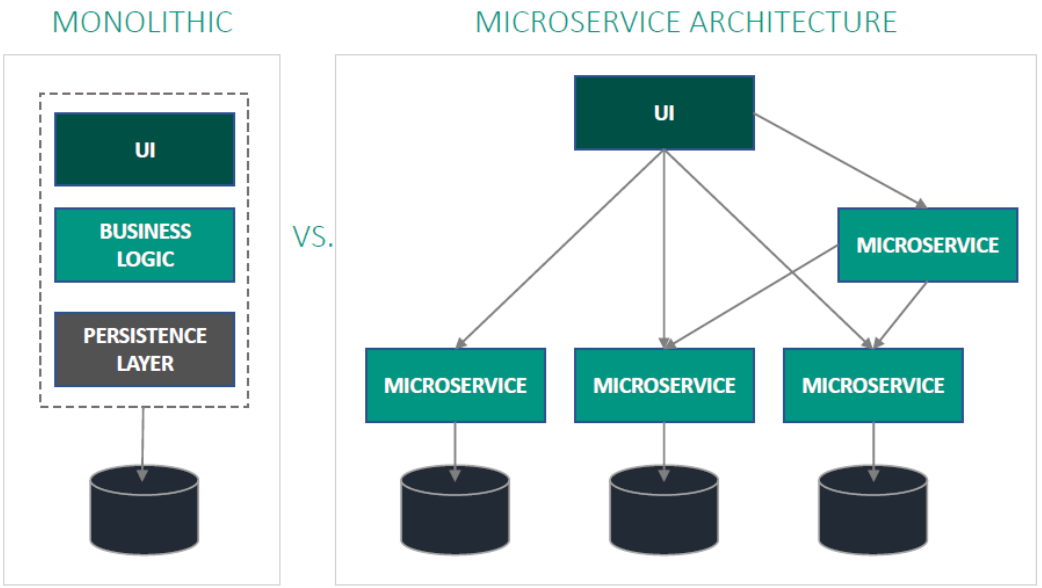
Supervisor: Dr. rer. nat. Robert Heinrich
Student: Niko Benkler

SOFTWARE DESIGN AND QUALITY GROUP
INSTITUTE FOR PROGRAM STRUCTURES AND DATA ORGANIZATION, FACULTY OF INFORMATICS



Microservice Architecture

- Benefits
  - Availability, Resilience, Fault Isolation
  - Scalability, Resource Utilisation
  - Neutral Development Technology
- Challenges
  - Expensive Communication
  - Organizational Challenges
  - **Microservice Identification**



PIBA

Problem

- Identifying microservices is a cumbersome task and requires detailed knowledge and manual effort
- Strong dependencies in control flow and data flow make identification difficult

Idea

- Identify microservices using clustering on control flow and data flow

Benefit

- Reduce required expertise and manual effort
- Create adequate microservices

Action

- Extract control flow (activities) and data flow (data object) dependencies from BPMN models
- Create two weighted graphs based on the dependencies
- Identify highly cohesive sets of activities and data objects
- Match clusters to generate microservice candidates

Research Questions

- **RQ1**: How to identify microservices based on the system specifications?
- **RQ1.1**: Which is an appropriate strategy to decompose a system into microservices?
  - Literature research to find suitable strategies
- **RQ1.2**: How to identify possible microservices without detailed knowledge and manual effort?
  - Most promising strategy identified in **RQ1.1** as basis
  - Elaborate approach
- **RQ1.3**: What is the accuracy of the approach?
  - Comparing the results of the approach with results of other approaches

Literature Research

- Conducted using digital libraries
  - *IEEE*
  - *ACM*
  - *SpringerLink*
  - *Google Scholar* for further information
- Eight promising approaches were found
- Approaches compared using eight defined criteria

["identify" OR "identification" OR "migrating" OR "monolith" OR "decomposition" OR "decompose monolith" OR "decompose"] AND "microservice"
OR
"microservice" AND ["identification" OR "transformation" OR "refactor"]

State of the Art I

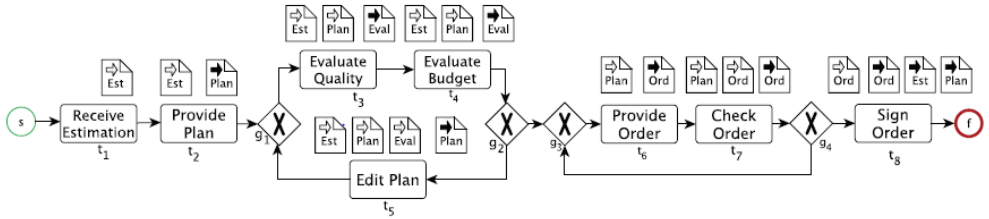
Approach/Criterion	Munezero et al. [22]	Chen et al. [5]	Alwis et al. [6]	Gysel et al. [11]
Basic Concept	define business capabilities by using domain-driven design patterns	algorithmic identification of microservices using data flows	graph-based identification process using heuristics to describe call graph similarities	service decomposition based on 16 coupling criteria
Prerequisites	domain defined by ubiquitous language	systems's data flows constructen on users' natural langugae description	Log files of legacy system	various System Specification Artifacts (SSAs) in specified format
Input	well defined domain model	Data Flow Diagrams (DFD)	Call Graphs, Source Code, System Database	instances of SSAs (e.g. ERM models, use cases)
Tool support	n/a	n/a	External tool for generating call graphs	implementation and wiki available
Degree of human involvement	domain experts define boundaries for business responsibilities	manual construction of purified DFD	no interaction needed	priorization of coupling criteria
Granularity	depends on the size of the defined business capability	most fine-grained ms candidates in terms of data operations	lowest granularity of sw based on structural and behavioural properties	n/a
Validation	demonstrated on sample domain	two case studies verified against relevant microservice principles and results of [11]	two experiemtns with complex enterprise systems (legacy vs. ms implementation)	validation via implementation and two case studies
Limitation	only conceptional approach, requires vast amount of expertise	transforming purified DFD not trivial (identifying same data operations requires expertise)	requires expressive log files to generate call graphs and identify business object relationships	generating SSAs in specified format is work intense

State of the Art II

Approach/Criterion	Mazlami et al. [20]	Amiri [2]	Baresi et al. [3]	Tyszberowicz et al. [24]
Basic Concept	meta-data aided graph clustering	business process oriented graph clustering	semantic similarity of OpenApi specification	functional decomposition of sw requirements
Prerequisites	applications with meaningful VCS data	business processes and entities available	well-defined Api with proper naming	specification of software requirements
Input	Source Code and VCS meta data	BPMN business processes with data object reads and writes	reference vocabulary (fitness function), OpenApi specifications	use cases
Tool support	prototype available (https://github.com/gmaz frontend)	Clustering tool "Bunch"	experimental prototype (https://github.com/mgar riga/decomposer)	use external graph visualize and analyse tools
Degree of human involvement	choose amount of clusters that will represent the microservices	no interaction needed	user defines level of hierachy	manual elimination of synonyms, irrelevant nouns and verbs
Granularity	depends on choosen amount of clusters	depends on iteration of genetic algorithm for convergence of fitness function	depends on choosen hierachy lebel, varies from one to many	depends on size of business capability
Validation	experiements using open-source projects with VCS data (200 to 25000 commits, 1000 to 500000 LOC, 5 to 200 authors)	multiple experiments, results compared with domain experts knowledge	452 OpenApi specification, 5 samples compared with results of sw-engineers and [11]	case study, compared to three manual implementations
Limitation	need meaningful VCS data and ORM model for its data entities	given weight definitions lack formal explanation	depends on reference vocabulary and well-defined interfaces	manual revision of operations (nouns) and state variable (verbs)

Inspired by "*Object-aware Identification of Microservices*" [1]

- BPMN Graph:



- Structural Dependency

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
t_1	0	1	0	0	0	0	0	0
t_2	0	0	1	0	0	0	0	0
t_3	0	0	0	1	0	0	0	0
t_4	0	0	0	0	1	1	0	0
t_5	0	0	1	0	0	0	0	0
t_6	0	0	0	0	0	0	1	0
t_7	0	0	0	0	0	1	0	1
t_8	0	0	0	0	0	0	0	0

- Data Object Dependency

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
t_1	0	1/4	1/4	1/4	1/4	0	0	1/2
t_2	1/4	0	3/4	3/4	5/4	1/2	1/2	3/2
t_3	1/4	3/4	0	3/2	5/4	1/4	1/4	1
t_4	1/4	3/4	3/2	0	5/4	1/4	1/4	1
t_5	1/4	5/4	5/4	5/4	0	1/2	1/2	3/2
t_6	0	1/2	1/4	1/4	1/2	0	5/4	3/2
t_7	0	1/2	1/4	1/4	1/2	5/4	0	3/2
t_8	1/2	3/2	1	1	3/2	3/2	3/2	0

- Aggregation

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
t_1	0	5/4	1/4	1/4	1/4	0	0	1/2
t_2	1/4	0	7/4	3/4	5/4	1/2	1/2	3/2
t_3	1/4	3/4	0	5/2	5/4	1/4	1/4	1
t_4	1/4	3/4	3/2	0	9/4	5/4	1/4	1
t_5	1/4	5/4	9/4	5/4	0	1/2	1/2	3/2
t_6	0	1/2	1/4	1/4	1/2	0	9/4	3/2
t_7	0	1/2	1/4	1/4	1/2	9/4	0	5/2
t_8	1/2	3/2	1	1	3/2	3/2	3/2	0

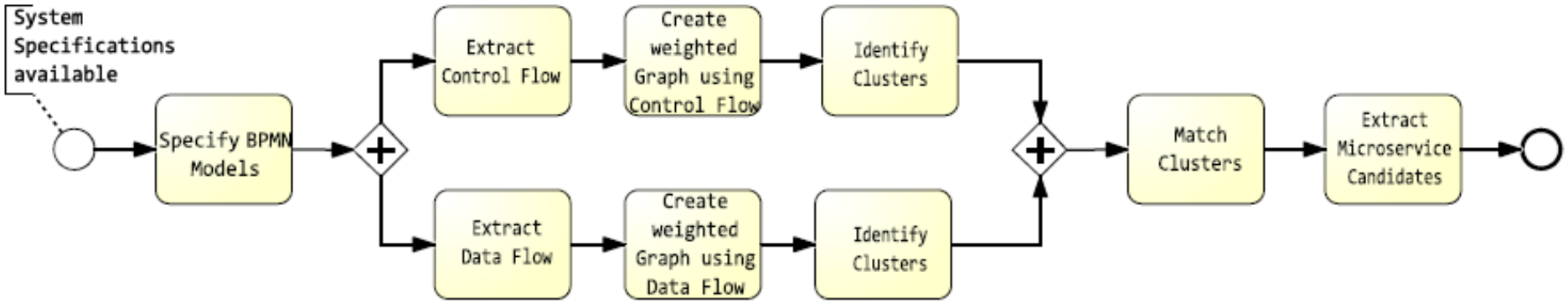
Clustering

Microservice candidates

Sources: Object-aware Identification of Microservices, M.J. Amiri [1]

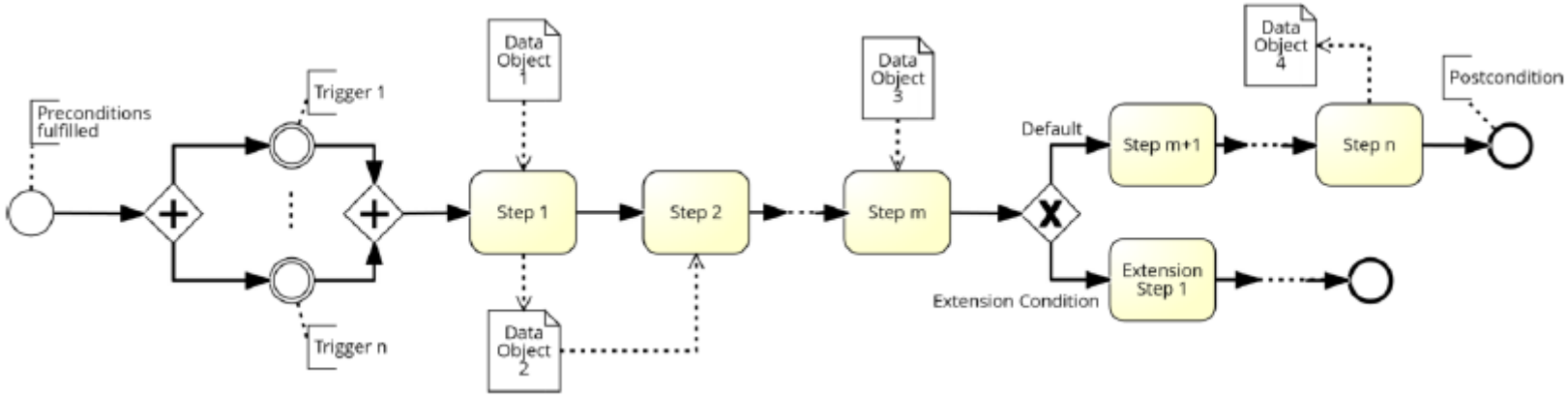
Approach

- Divided in nine steps



Specify BPMN Models

- System requirements in various forms
- Transformation into BPMN models
  - Workshops
  - Use Case Transformation
  - Others: BPMN Miner[5] …

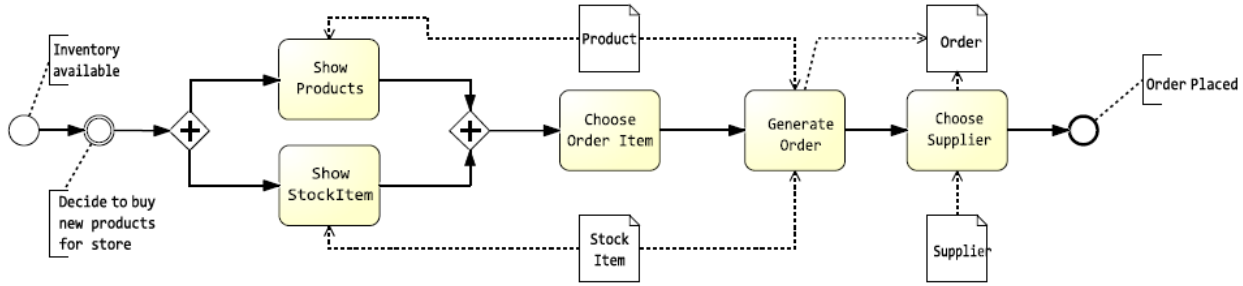


Use Case Transformation based on [2]

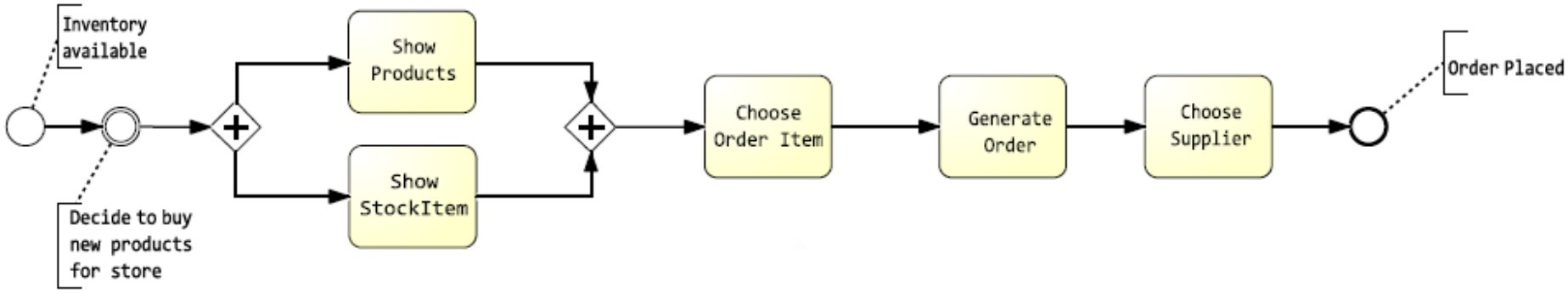
Extract Control Flow

- Delete Data Objects and accompanying associations

Original:



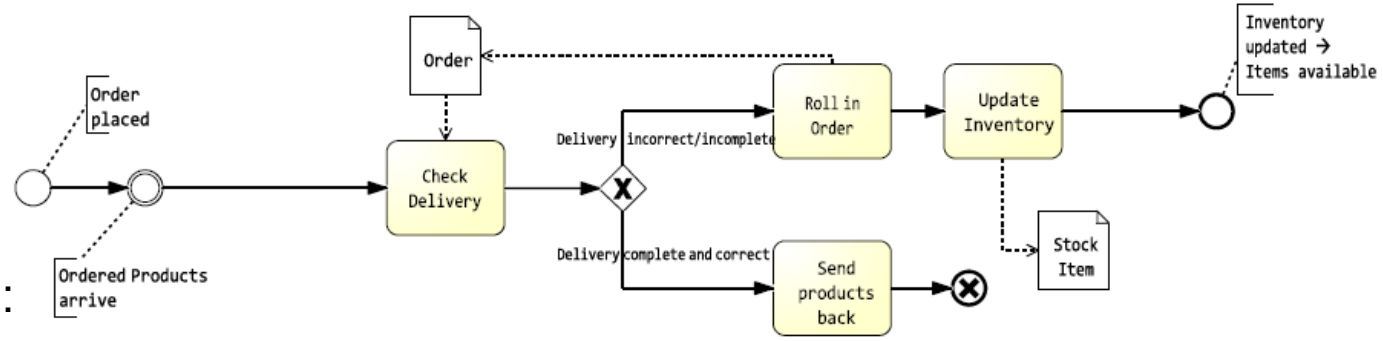
Extracted Control Flow:



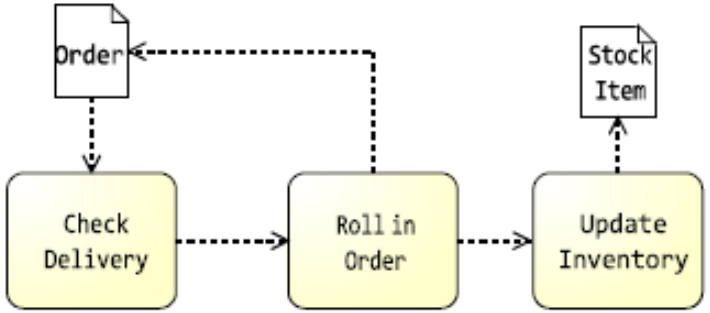
Extract Data Flow

- Approximate Data Flow
- Delete control flow related parts
- Connect pair of tasks if connected by control flow arc
- Replace Gates
  - Replace by two data flow arcs
  - No distinction between XOR and parallel Gateway
- Delete unnecessary tasks

Original:

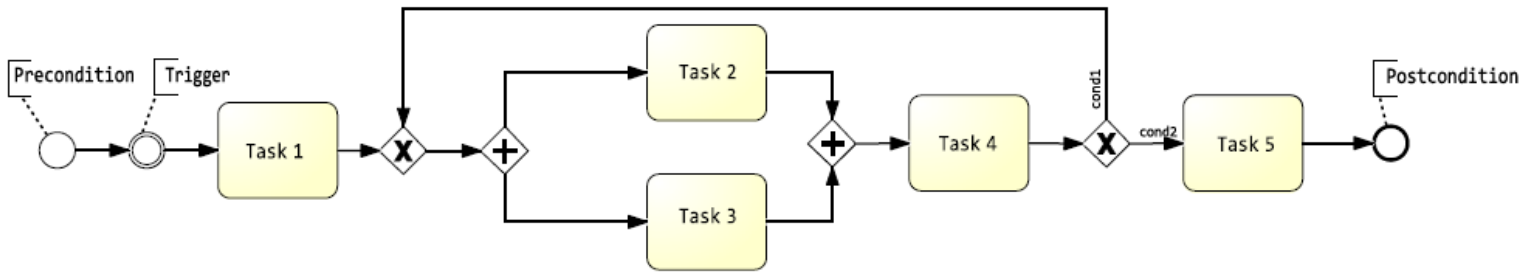


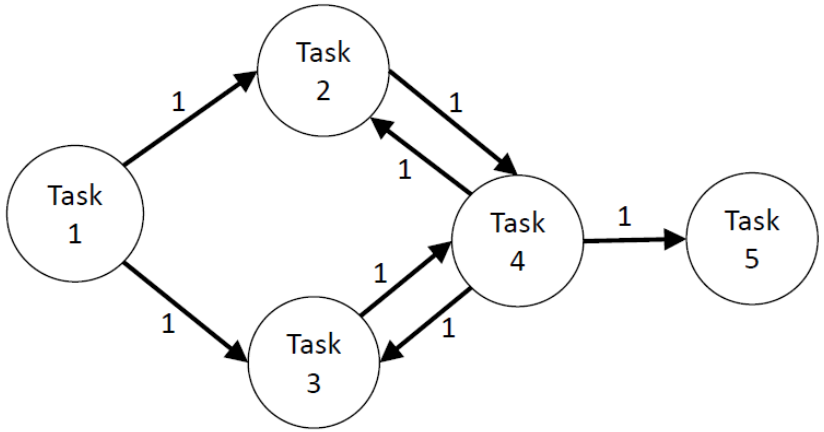
Extracted Data Flow:



Create a weighted Graph using Control Flow

- Generate one Graph by using all control flow models
- Connect pair of activities
  - if directly connected in BPMN models
  - if only gateways in between
- Assign a weight of **1** to all dependencies
- Multiple occurrences: Add weights



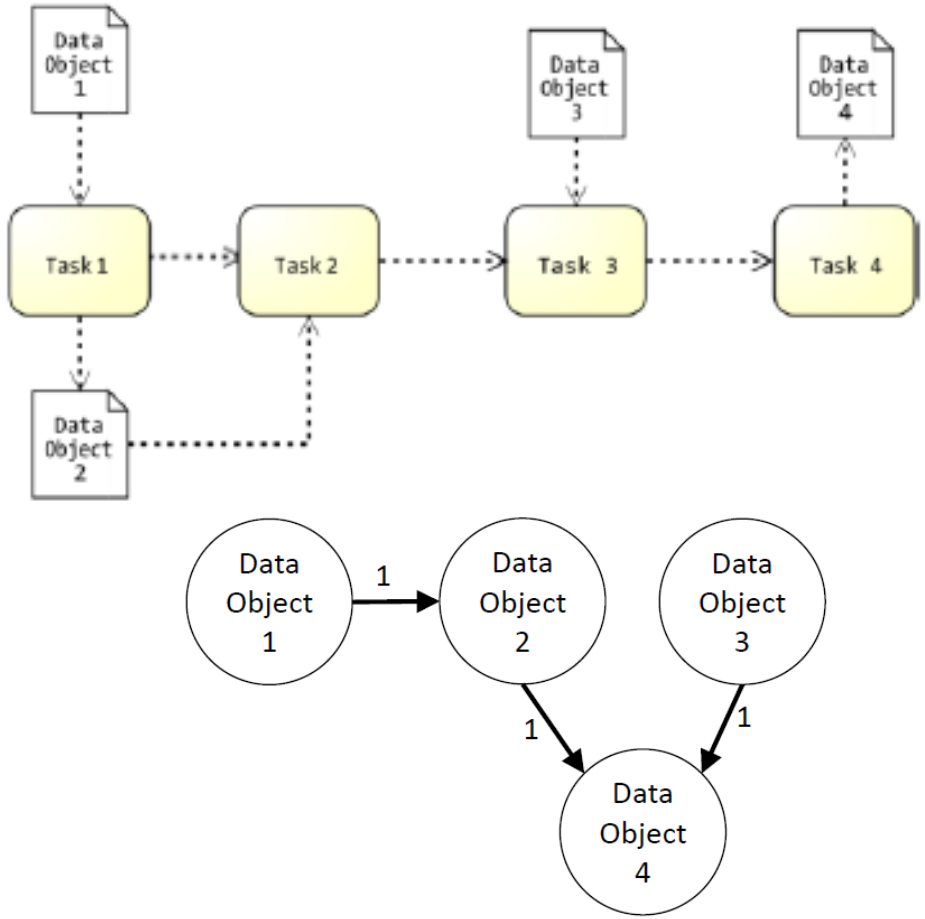


Create a weighted Graph using Data Flow

- Generate one Graph by using all data flow models
- Connect pair of data objects
  - if both data objects are read by the same task
  - if a data object is used to update another data object

→ max. ***n*** tasks in between a task that reads the first data object and another tasks that updates the other data object

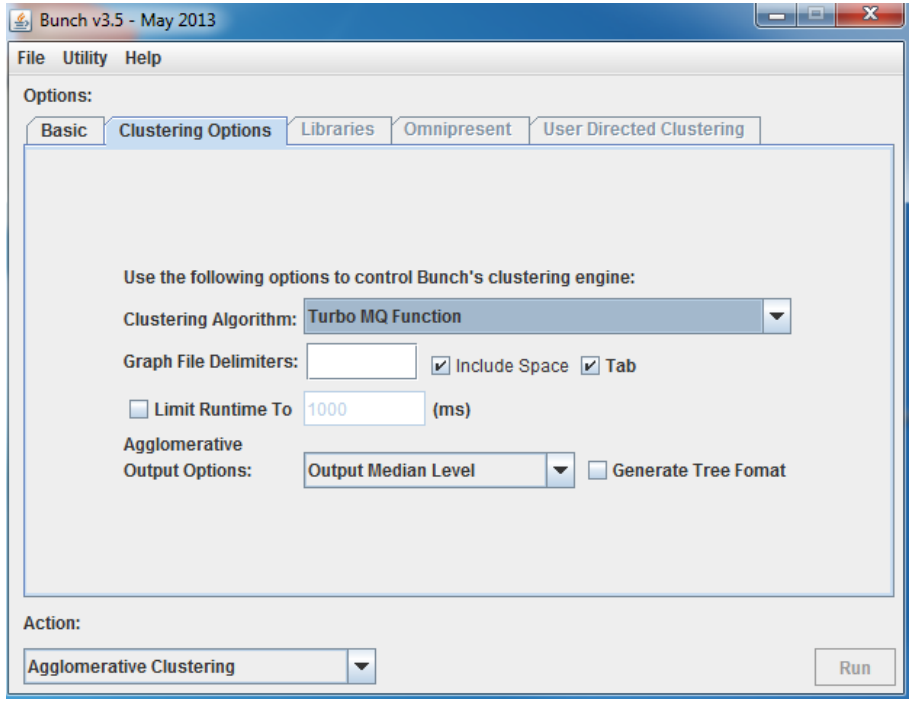
- Determine parameter ***n*** depending on the granularity of the *BPMN* models
- Assign a weight of **1** to all dependencies
- Multiple occurrences: Add weights



Identify Clusters

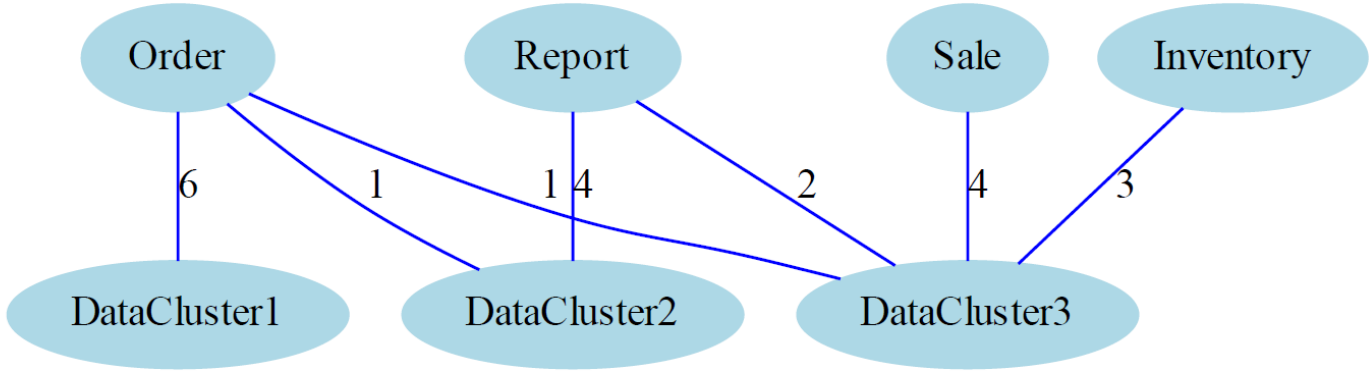
- *Bunch Software [3]*
  - Genetic Algorithm: Randomly picks k cluster
  - Fitness Function: *Turbo-MQ*
  - Cluster Factor: Rewards intra-cluster coupling
- Input: List of edges with weights
- Output: *DOT* format
- Visualization: *GraphViz* [4]
- Two sets of cluster
  - Activity cluster
  - Data Object Cluster

$$Turbo - MQ = \sum_{i=1}^{k} CF_i \qquad CF_i = \begin{cases} 0 & \mu_i = 0 \\ \frac{\mu_i}{\mu_i + \epsilon_i} & otherwise \end{cases}$$



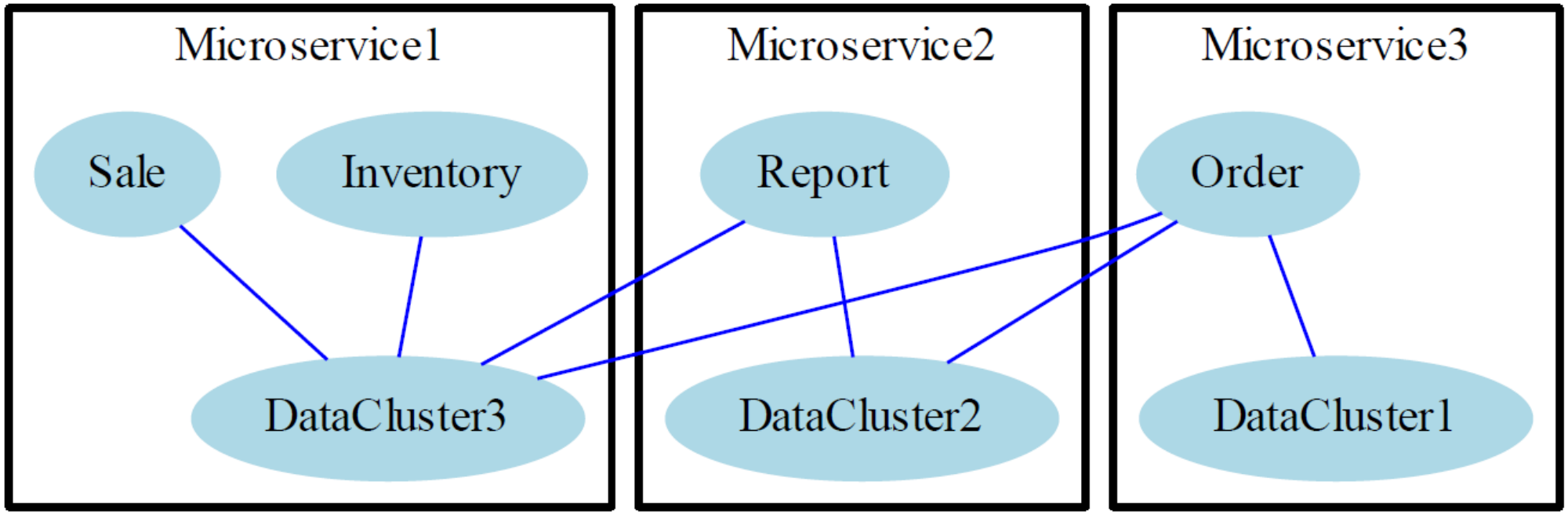
Matching of Clusters

- Count data object access between activity clusters and data object cluster
- Use amount as weight
- Use *Bunch* to identify compound cluster



Extract Microservice Candidates

- Each compound cluster correspond to a microservice candidate

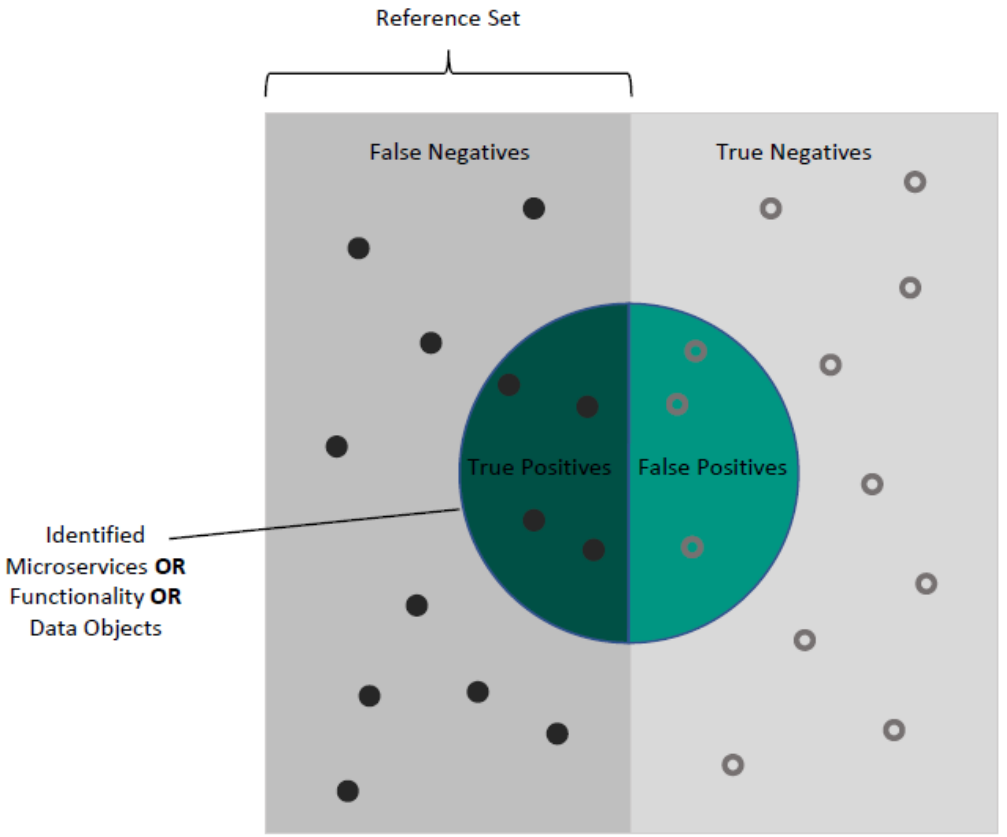


Evaluation

- Goal: Determine the accuracy of the approach
- Comparison to two Reference Sets
- Questions: What is the *Precision and Recall* regarding the identified
  - microservices?
  - functionalities of the microservices?
  - data objects of the microservices?

$$Precision = \frac{N_{ret \cap rel}}{N_{ret}} \qquad Recall = \frac{N_{ret \cap rel}}{N_{rel}}$$

	Relevant	Not Relevant	Sum
Retrieved	$N_{ret \cap rel}$	$N_{ret \cap \overline{rel}}$	N_{ret}
Not Retrieved	$N_{\overline{ret} \cap rel}$	$N_{\overline{ret} \cap \overline{rel}}$	$N_{\overline{ret}}$
Sum	N_{rel}	$N_{\overline{rel}}$	N_{total}



Evaluation Design

- *CoCoME* as running example
  - Community case study for software evolution
  - Typical component-based information system
- Comparison to two reference sets
  - Well established procedure to reason about the accuracy
- *Precision and Recall* capable to determine accuracy of the approach
  - How many relevant instances identified
  - How many of the identified are correct
- Reference Sets
  - Decomposition by approach in “Identifying Microservices Using Functional Decomposition” [4]
  - Manual Decomposition

Evaluation Results

- Reference Set 1 does not contain all functionalities

  ➡ focus is on Reference Set 2 (Manual Decomposition)

- No *false positive* microservices
- $\text{Recall}_{\text{microservice}} = \frac{3}{4} = 0.75$ $\text{Precision}_{\text{microservice}} = \frac{3}{3} = 1$
- $\text{Recall}_{\text{functionality}} = \frac{12}{18} \approx 0.67$ $\text{Precision}_{\text{functionality}} = \frac{12}{13} \approx 0.92$
- $\text{Recall}_{\text{dataObject}} = \frac{5}{7} \approx 0.71$ $\text{Precision}_{\text{dataObject}} = \frac{5}{7} \approx 0.71$

➡ Satisfying results

Conclusion

- Approaches exist, but no one considers control and data flow together
- Contributions:
  - Strategy developed that fills the gap in current research
  - Approach elaborated for identifying microservices using clustering on control flow and data flow
- Evaluation:
  - (Semi-)automatic approach capable of identifying microservices
  - Reduces effort and necessary knowledge
  - Produces adequate microservices in the case of CoCoME

Limitations and Future Work

- Limitations
  - Transformation of system specifications into BPMN models not trivial
  - Same granularity for all BPMN models required
  - Parameter *n* needs knowledge about granularity

- Future Work
  - Additional data flow diagram needed?
  - Different clustering algorithms to achieve variable microservice sizes
  - Approach capable of identifying different microservice sizes?
  - Cluster matching: Elaborate *white box* approach
  - Apply on other systems

Bibliography

[1] M. J. Amiri. “Object-Aware Identification of Microservices”. In: (July 2018), pp. 253–256

[2] D. Lubke, K. Schneider, and M. Weidlich. “Visualizing Use Case Sets as BPMN Processes”

[3] Bunch Software, https://www.cs.drexel.edu/~spiros/bunch/, Accessed on 15.04.2019

[4] Shmuel Tyszberowicz et al. “Identifying Microservices Using Functional Decomposition”

[5] Raffaele Conforti et al. “BPMN Miner: Automated discovery of BPMN processmodels with hierarchical structure”


IT'S FINISHED.
IT'S DONE.
quickmeme.com

PIBA

Problem

- Identifying microservices is a cumbersome task and requires detailed knowledge and manual effort
- Strong dependencies in control flow and data flow make identification difficult

Idea

- Identify microservices using clustering on control flow and data flow

Benefit

- Reduce required expertise and manual effort
- Create adequate microservices

Action

- Extract control flow (activities) and data flow (data object) dependencies from BPMN models
- Create two weighted graphs based on the dependencies
- Identify highly cohesive sets of activities and data objects
- Match clusters to generate microservice candidates