Service Cutter: A Systematic Approach to Service Decomposition

Auteurs: **Michael Gysel, Lukas Kölbener, Wolfgang Giersche** et **Olaf Zimmermann**

University of Applied Sciences of Eastern Switzerland & Zühlke Engineering

2016

Guillaume André, Marc Jourdes, Jean-Adam Puskaric

Objectifs

- Fournir un **catalogue de critères de couplage** pour la décomposition en services.

- Construire une méthode de décomposition systématique en services basée sur les **besoins du domaine métier**, partiellement **automatique**, **répétable** et capable de **passer à l'échelle.**

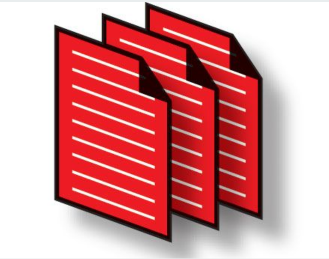
- **Assister** les décisions prises sur les architectures de services.

Scénarios d’utilisation

Approche **Green Field**

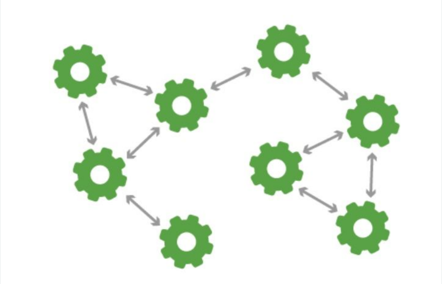
Approche **Monolithe**

Fonctionnement général



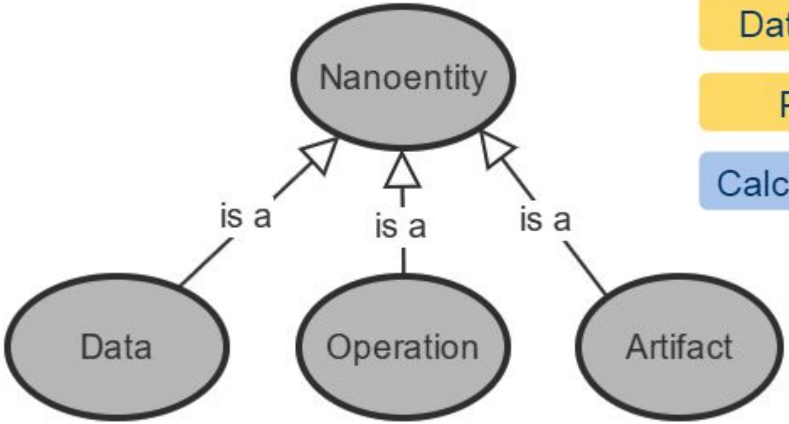
Spécifications

Service Cutter



Proposition d'architecture

Nanoentités



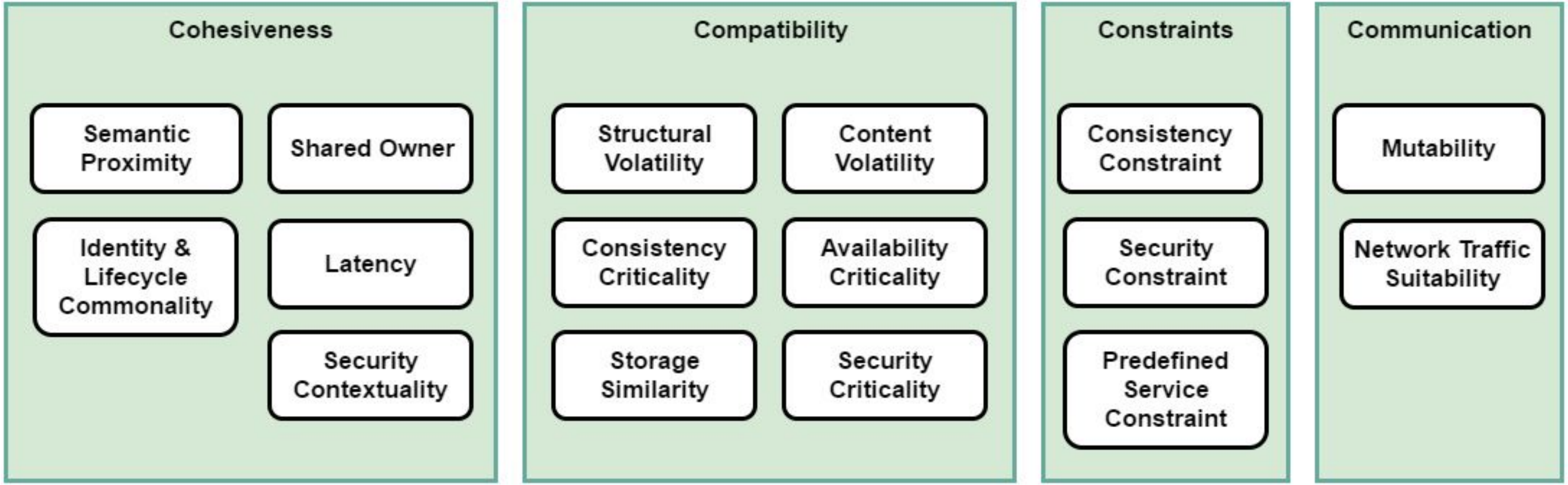
Name	Birthday
DateOfEmployment	Address
PhoneNumber	GrossSalary
CalculatePayrollTaxes	PaymentSlip

🟨 = Data

🟦 = Operation

🟩 = Artifact

Critères de couplage



Cohesiveness

- Semantic Proximity
- Shared Owner
- Identity & Lifecycle Commonality
- Latency
- Security Contextuality

Compatibility

- Structural Volatility
- Content Volatility
- Consistency Criticality
- Availability Criticality
- Storage Similarity
- Security Criticality

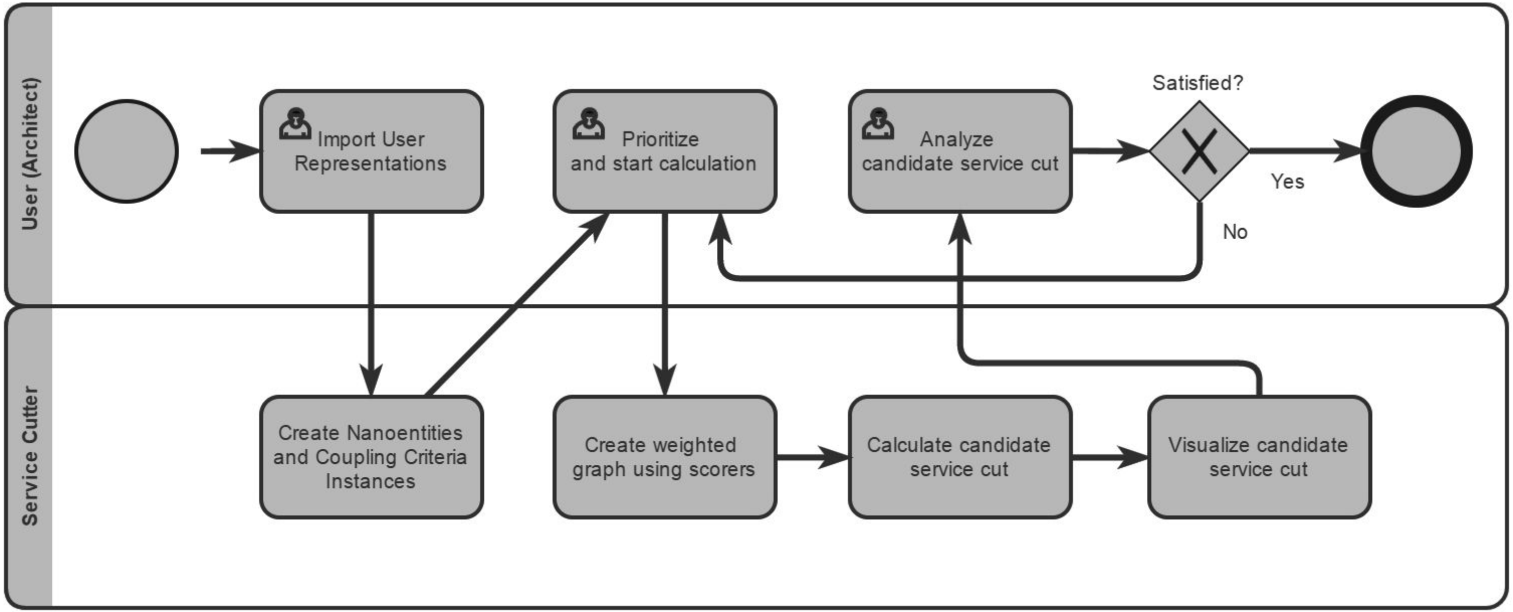
Constraints

- Consistency Constraint
- Security Constraint
- Predefined Service Constraint

Communication

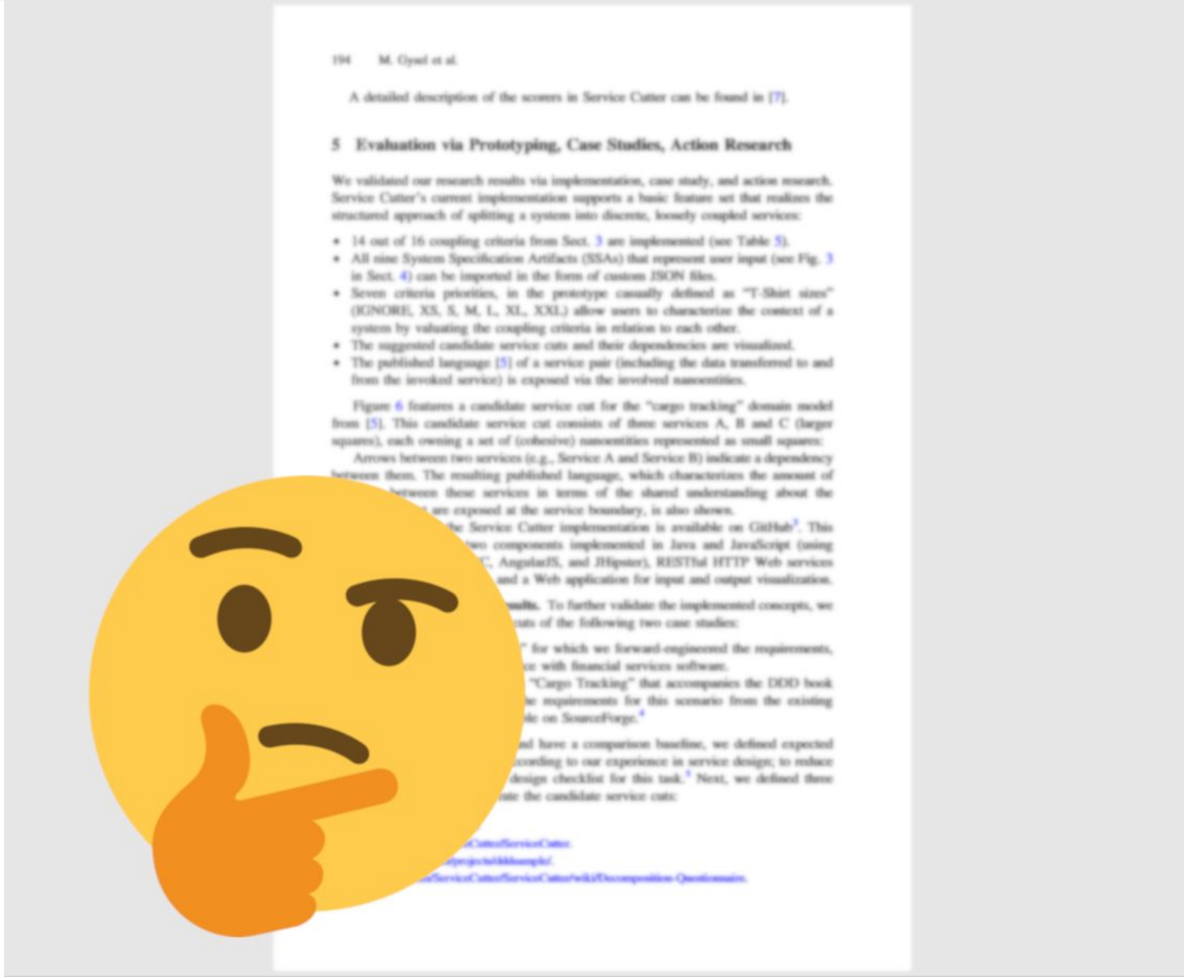
- Mutability
- Network Traffic Suitability

Fonctionnement du Service Cutter


User (Architect)
Import User Representations
Prioritize and start calculation
Analyze candidate service cut
Satisfied?
Yes
No
Service Cutter
Create Nanoentities and Coupling Criteria Instances
Create weighted graph using scorers
Calculate candidate service cut
Visualize candidate service cut

Résultats et critiques

L’identification de service c’est pas automatique !



194 M. Gysel et al.

A detailed description of the scorers in Service Cutter can be found in [7].

5 Evaluation via Prototyping, Case Studies, Action Research

We validated our research results via implementation, case study, and action research. Service Cutter’s current implementation supports a basic feature set that realizes the structured approach of splitting a system into discrete, loosely coupled services:

- 14 out of 16 coupling criteria from Sect. 3 are implemented (see Table 5).
- All nine System Specification Artifacts (SSAs) that represent user input (see Fig. 3 in Sect. 4) can be imported in the form of custom JSON files.
- Seven criteria priorities, in the prototype casually defined as “T-Shirt sizes” (IGNORE, XS, S, M, L, XL, XXL) allow users to characterize the context of a system by valuating the coupling criteria in relation to each other.
- The suggested candidate service cuts and their dependencies are visualized.
- The published language [5] of a service pair (including the data transferred to and from the invoked service) is exposed via the involved nanoentities.

Figure 6 features a candidate service cut for the “cargo tracking” domain model from [5]. This candidate service cut consists of three services A, B and C (larger squares), each owning a set of (cohesive) nanoentities represented as small squares:

Arrows between two services (e.g., Service A and Service B) indicate a dependency between them. The resulting published language, which characterizes the amount of
between these services in terms of the shared understanding about the
are exposed at the service boundary, is also shown.
the Service Cutter implementation is available on GitHub[3]. This
two components implemented in Java and JavaScript (using
C, AngularJS, and JHipster), RESTful HTTP Web services
and a Web application for input and output visualization.

sults. To further validate the implemented concepts, we
cuts of the following two case studies:

” for which we forward-engineered the requirements,
ce with financial services software.
“Cargo Tracking” that accompanies the DDD book
he requirements for this scenario from the existing
ble on SourceForge.[4]

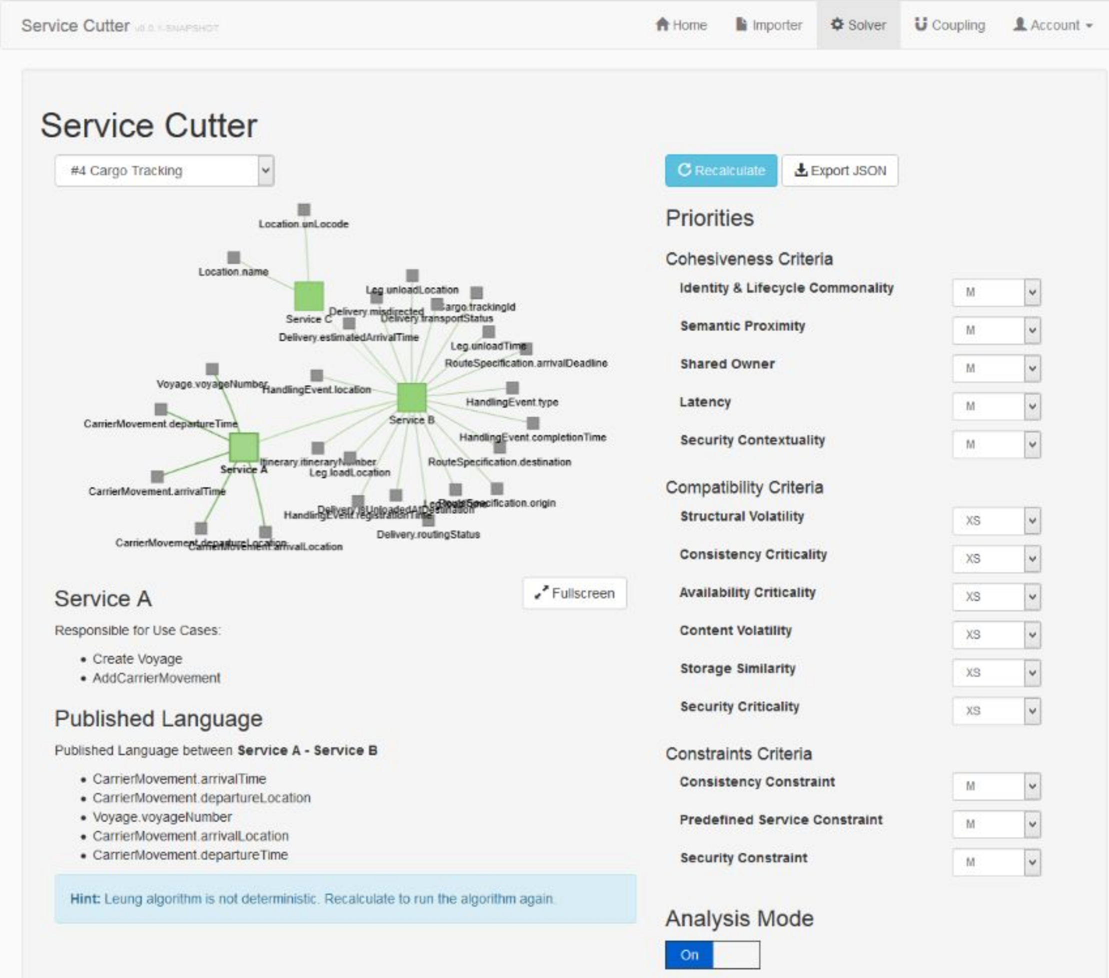
nd have a comparison baseline, we defined expected
ccording to our experience in service design; to reduce
design checklist for this task.[5] Next, we defined three
ate the candidate service cuts:

Cutter/ServiceCutter.
/projects/dddsample/.
/ServiceCutter/ServiceCutter/wiki/Decomposition-Questionnaire.

L’outil

- + Facile d’utilisation
- + Bien documenté
- + L’approche nanoentité
- + Rapide et pramétrisable

- - Demande beaucoup de paramètres
- - Résultats pas toujours cohérents
- - Pas d’extraction de modèle automatique


Service Cutter v0.0.1-SNAPSHOT
Home
Importer
Solver
Coupling
Account
Location.unLocode
Location.name
Service C
Leg.unloadLocation
Delivery.misdirected
Cargo.trackingId
Delivery.transportStatus
Delivery.estimatedArrivalTime
Leg.unloadTime
RouteSpecification.arrivalDeadline
Voyage.voyageNumber
HandlingEvent.location
Service B
HandlingEvent.type
CarrierMovement.departureTime
HandlingEvent.completionTime
Service A
Itinerary.itineraryNumber
Leg.loadLocation
RouteSpecification.destination
CarrierMovement.arrivalTime
Delivery.isUnloadedAtDestination
RouteSpecification.origin
HandlingEvent.registrationTime
Delivery.routingStatus
CarrierMovement.departureLocation
CarrierMovement.arrivalLocation
Fullscreen
Recalculate
Export JSON

Service Cutter

#4 Cargo Tracking

Service A

Responsible for Use Cases:

- Create Voyage
- AddCarrierMovement

Published Language

Published Language between **Service A - Service B**

- CarrierMovement.arrivalTime
- CarrierMovement.departureLocation
- Voyage.voyageNumber
- CarrierMovement.arrivalLocation
- CarrierMovement.departureTime

Hint: Leung algorithm is not deterministic. Recalculate to run the algorithm again.

Priorities

Cohesiveness Criteria

Identity & Lifecycle Commonality	M
Semantic Proximity	M
Shared Owner	M
Latency	M
Security Contextuality	M

Compatibility Criteria

Structural Volatility	XS
Consistency Criticality	XS
Availability Criticality	XS
Content Volatility	XS
Storage Similarity	XS
Security Criticality	XS

Constraints Criteria

Consistency Constraint	M
Predefined Service Constraint	M
Security Constraint	M

Analysis Mode

On

L’identification de service, un challenge de la recherche