Semantic Composition of Optimal Process Service Plans in Manufacturing with ODERU

Luca Mazzola*
HSLU, Hochschulke Luzern
Informatik
Campus Zug-Rotkreuz, Suurstoffi 41b
CH-6343, Risch-Rotkreuz, Switzerland
luca.mazzola@hslu.ch
mazzola.luca@gmail.com

Patrick Kapahnke
DFKI, German Research Center for AI
Agents and Simulated Reality
Saarland Informatics Campus D3.2
D-66123, Saarbrücken, Germany
p.kapahnke@evana.de

Matthias Klusch
DFKI, German Research Center for AI
Agents and Simulated Reality
Saarland Informatics Campus D3.2
D-66123, Saarbrücken, Germany
klusch@dfki.de

ABSTRACT

Purpose

The need to flexibly react to changing demands and to cost-efficiently manage customized production even for unitary batch size, requires a dynamic and holistic integration of service-based processes within and across enterprises of the value chain. In this context, we present a novel pragmatic approach called ODERU for automatically implementing service-based manufacturing processes at design and runtime within a cloud-based elastic manufacturing platform.

Design/methodology/approach

ODERU relies on a set of semantic annotations of business process models encoded into an extension of the BPMN 2.0 standard. Leveraging the paradigms of semantic SOA and XaaS, ODERU integrates pattern-based semantic composition of process service plans with QoS-based optimization based on multi-objective COP solving.

Findings

The successful validation of ODERU in two industrial use cases for maintenance process optimization and automotive production in the European project CREMA revealed its usefulness for service-based process optimization in general and for significant cost reductions in maintenance, in particular.

Originality/value

ODERU provides a pragmatic and flexible solution to optimal service composition with three main advantages: (1) Full integration of semantic service selection and composition with QoS-based optimisation; (2) Executability of the generated optimal process service plans by an execution environment as they include service assignments, data flow (variable bindings) and optimal variable assignments; and (3) Support of fast replanning thanks to the storage into a single location for model and plan.

Keywords: Process optimization, semantic business process management, BPMN, semantic SOA, service orchestration, Industry 4.0
Paper type: Research paper

1 INTRODUCTION

About a decade ago, the fourth industrial revolution, also known as Industry 4.0, has been ushered by the introduction of the Internet of Things and Services into the manufacturing environment. Industry 4.0 is focused on creating smart products and processes flexibly in dynamic, real-time optimised and self-organising value chains, and profitably even down to production lot size of one. To rise up to this challenge, Industry 4.0 applications basically operate on the principles and use of autonomous cyber-physical systems with self-configuring properties for integrated production across the entire value chain. In particular, the IP-networked and sensor-equipped machinery, systems, vehicles and devices of smart factories are vertically and horizontally integrated with service-based business processes both within a company and inter-company value networks. Besides, cyber-physical production systems are envisioned to not only cooperate with each other but also with humans on a new level of socio-technical interaction. From a holistic perspective, Industry 4.0 connects smart production closely with the areas of smart transport and logistics, smart buildings, and smart energy, while keeping humans in the loop via smart multimodal assistance in modern working environments.

The envisioned integration of optimal service-based processes within and across the enterprise of dynamic value chains requires, in particular, smart tool support for an automated generation of process service plans that are optimal with respect to both, functional and non-functional QoS-based requirements at design time and runtime. In addition, the provided process service plans (PSP) should be generated in a way that supports an effective re-planning at runtime in case an included service is temporarily failing or becomes unavailable. That goes beyond the capability of conventional BPM (business process modeling) systems.

To this end, we developed a novel pragmatic solution called ODERU (Optimization tool for DEsign and RUntime). ODERU computes the set of functionally optimal process service plans based on semantic annotations of executable services and process models, followed by the computation of top-k fully optimal process service plans based on the solving of the embedded COP (constrained optimisation problem) of the process model in extended BPMN. The resulting complete and executable optimal process service plan including the required variable bindings and the environmental variables assignment is encoded back into specifically developed BPMN 2.0 extensions, thereby bridging the gap between process models and executable process service plans.

The remainder of this paper is structured as follows. Section II presents related work, followed by the description of the ODERU solution in terms of the overall architecture, methods and interfaces, and an estimation of its complexity in Section III. Section IV provides an illustrating simple example of using ODERU for

*Dr. Mazzola worked at DFKI during the CREMA project and ODERU development.

optimal composition of process service plans. Section V presents a summary of the use and validation of ODERU in two real-world industrial use cases in the domain of Industry 4.0 that have been adopted in the European project CREMA, and Section VI concludes the paper.

2 RELATED WORK

At the core, ODERU follows the paradigm of Semantic Service-Oriented Architectures (SemSOA). Process models are automatically implemented with semantic services by applying techniques of semantic service selection and composition planning. The key idea is to enable automated understanding of task requirements and services by providing semantic descriptions in a standardized machine-understandable way by using formal ontological definitions [1], for example in OWL2 [2].

To apply this paradigm to business processes, several initiatives and approaches exist and reference architectures as well as frameworks for semantic business process modeling are proposed in literature. In [3], the benefit of adding semantics to BPM (SBPM) is discussed, in particular focusing on the modeling and configuration phases. They propose to make use of semantics to support the modeling in terms of service selection and composition on task level and by means of semantic validation, which enables consistency checks of effects (e.g. for parallel execution) among others. A more detailed investigation of this aspect can be found in [4]. Similarly, service bindings can be found during configuration using semantic annotations. The authors base their methodology on BPMN, BPEL and WSMO. Along the same lines, the authors of [5] propose a similar SBPM framework, which combines semantic web services and BPM to overcome the problem of automated understanding of processes by machines in a dynamic business environment. The idea is to make use of WSMO in addition to standard BPMN to represent the semantics of a business process and its parts. While both works solve the issue of semantic understanding and provides rationale on the benefit of SBPM, there is no integration into existing standards and multiple representations have to be maintained separately. Similarly, the authors of [6] propose sBPMN, which integrates semantic technologies and BPMN to overcome the obvious gap between abstract representation of process models and actual executable descriptions in BPEL. In particular, they propose an ontology, which is supposed to capture all the required semantic information. While this integrates both views, sBPMN is not suited to be used by existing BPMN tools without additional transformation. [7] follows the same track with the proposal of BPMO, an ontology, which partly is based on sBPMN, while [8] takes sBPMN as basis for the Maestro tool, which implements the realisation of semantically annotated business tasks with concrete services by means of automatic discovery and composition. In [9], a reference architecture for semantic SOA in BPM is proposed, which aims to address the representation discrepancy business expertise and IT knowledge by making use of semantic web technologies. The authors highlight the benefit of this approach by showing capabilities emerging from this combination, like semantic process model composition and auto-completion of process models. Like the other approaches shown before, they do not propose an integrated formalism, but rely on their compiler-like framework and semantic plug-in concept to bridge the representation gap. All of these proposals rely on formalizations different from (although based on) BPMN or do not aim for a full integration from a formalism point of view. In contrast, ODERU proposes a set of BPMN extensions, which enable semantic interoperability in a semantic SOA as well as support process model composition, task service selection and process execution.

ODERU applies state of the art semantic service selection technologies [10] for implementing annotated process tasks. Typically, work on semantic service selection can be grouped in terms of the selection criteria and the employed matching approach. Functional service matchmaking considers the service signature (inputs, outputs; IO) and service specification (preconditions and effects; PE) [11]. Non-functional criteria, often referred to as quality of service (QoS) (e.g. costs, execution time, availability), can additionally be considered to find matching services in terms of functional *and* non-functional requirements [12–14]. A lot of work has been dedicated to improve on overall matching precision by not only making use of strict logic-based selection of services given formal descriptions of IOPE, but also text similarity metrics and structural computations or hybrid combinations thereof [15–17]. While showing very good results in terms of ranking precision, such approaches sacrifice the property of correctness with respect to the formal specifications as implied by logic-based reasoning. This is not feasible for ODERU, because it makes use of service selection as basis for a pattern-based functional process service plan composition. Therefore, ODERU employs a logic-based configuration of the iSeM matchmaker [18], which is capable of IOPE selection given formal semantic descriptions in OWL2 and PDDL [19]. Also, the QoS aspect will not be covered by the service selection component of ODERU directly. Instead, optimality in terms of non-functional QoS specifications is achieved on the process model level by solving (non-)linear multi-objective constraint optimization problems (COP) as an integrated follow-up to the pattern-based composition, which utilizes the service selection.

Most existing approaches to process service plan composition do not cover the combination of functional (semantic) aspects and non-functional (QoS-aware) optimization, but rather focus on one of them. Naturally, much effort has been put into the functional composition, because this is one of the basic requirements to compute executable plans. For example, [8, 20, 21] consider functional semantic annotations to implement business processes by means of a service composition plan. In contrast, some work focuses on optimizing process service plans with respect to QoS. [22] provides a survey giving an overview of existing approaches and initiatives in this direction and highlights research questions. In [23], a novel approach for QoS-aware workflow optimization is presented, which takes structural orchestration components such as AND, XOR, OR as well as loops and unstructured components into account. The optimization is performed by means of Integer Linear Programming, after a transformation from a non-linear problem to a linear one. Although the approach can extend to arbitrary QoS types, structurally complex and non-linear problems like solved by ODERU can not be tackled appropriately. Integrated functional and non-functional optimization has rarely been considered. One notable exception is the work presented in [24], which also claims that existing methods are restricted to predefined functionally valid

plan options. To overcome this, the authors present an integrated SAT-based cost planning solver, which takes logical reasoning and temporal planning into account, while at the same time optimizing QoS respecting a set of global constraints. While composition typically includes the computation of possible data flows, ODERU additionally finds optimal service variable assignments that are also required for executing the resulting plans. This is a novel feature not yet considered by existing work. Moreover, ODERU performs re-optimization of process service plans at runtime upon request by the runtime environment and based on information about the leasability of services, which is also a novel feature. Finally, ODERU employs means of RDF stream processing to react to service changes (non-functional QoS aspects) reported by the service registry. This information can be used to trigger optimizations proactively, if the RDF stream engine identifies that a previously computed process service plan is affected.

3 ODERU SYSTEM ARCHITECTURE

3.1 Overview

ODERU integrates semantic process service composition with QoS-aware plan optimisation for given annotated business process models in extended BPMN 2.0. That is in compliance with the paradigms of XaaS (Everything-as-a-Service) and SemSOA (semantic SOA). In particular, all available resources are assumed to be encapsulated as executable services which, in turn, are semantically annotated with a shared ontology in the standard ontology language OWL2. The overall goal of semantic service composition by ODERU is to assign semantic services to annotated process tasks in a process model such that the resulting process service plan (PSP) is optimal with respect to the given functional and QoS-based optimization constraints.

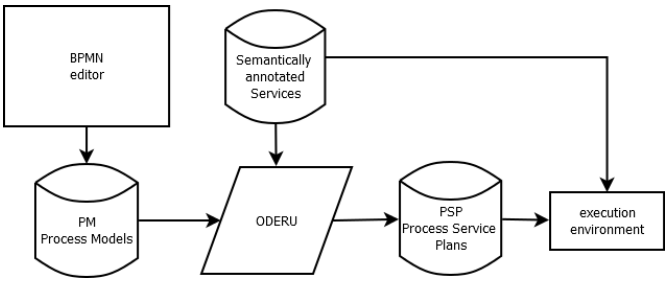

Figure 1: ODERU in a BPM and execution architecture

For an input process model (PM) in the semantically enriched extended BPMN format, ODERU computes an executable plan of services implementing the contained process tasks including information on the data flow between the services. In order to provide an optimal solution out of the computed set of possible functionally valid solutions, ODERU has to make particular choices driven by non-functional requirements, which are expressed as functions of the QoS measures provided by the services. Moreover, ODERU computes concrete settings of service input parameter values, which yield optimal results in terms of the optimisation criteria. Fig. 1 depicts the role of ODERU in the context of a business process modeling and execution application. A process designer specifies process models in BPMN using a graphical editor front end, that support the semantic annotation of IOPE (Input-Output-Precondition-Effect) aspects for each task. Process models are stored in a database which ODERU can access for its process service planning and optimisation. The generated executable process service plans are encoded in another BPMN 2.0 notation extension into the input PM, creating an instance of this process model, and stored in a repository which is accessible by a process execution environment. Services to be used for planning and later on execution are stored in a service repository. For details of the BPMN extensions used by ODERU, we refer to [25].

The incoming BPMN process models are expected to contain semantically annotated task descriptions as BPMN extension elements, which ODERU can map to logically equivalent or semantic plug-in matching services for execution. Analogous to the semantic service descriptions themselves, these annotations are structured in terms of IOPE and refer to domain knowledge specified in a shared ontology in OWL2. Moreover, the BPMN should specify which QoS measures have to be optimized and how they are defined. This is achieved by specifying a constrained optimization problem (COP) at the process model level, whose solutions dictate which services to select from a given set of alternatives and how to optimally set the parameters for executing these services. The COP formulation includes information on how to map optimal parameter values to service inputs and QoS parameters to COP constants. The outputs produced by ODERU are process service plans encoded in the original BPMN itself by making use of BPMN extensions again. Besides the optimal services and input values for calling the services as described above, this also includes possible data flows with parameter bindings among services. Such a process service plan implementing the process model can then be instantiated at runtime by a process plan execution environment under the following assumptions:

- Loop structures are unfolded during execution only, while ODERU assumes that service executions are non-exclusive in general (i.e. a single service can be called multiple times without any side effect). If a service *is* exclusive, the execution environment should trigger exception handling and ask back ODERU for a new plan implementing the rest of the process model with other equivalent or plug-in services.
- Gateways are handled by ODERU by computing data flow alternatives for each possible execution path depending on the gateway type. Each possible process execution flow is expressed inside a distinct process service plan. The execution environment should retrieve relevant alternatives from ODERU depending on how the gateways are evaluated.

To achieve this, ODERU works as follows in a sequential manner:

- Pattern-based composition using semantic service selection for all semantically annotated process tasks and computation of possible data flows.
- QoS-aware non-functional optimization by means of COP solving on the process model level. This step selects particular services out of sets of functionally fitting services per tasks and provides the optimal settings for service inputs.

This workflow can be applied at design time and runtime of a process model execution instance. At design time, ODERU will be called after a process model has been defined in order to provide an executable implementation of the model with services for the execution environment (cf. Fig. 1). The runtime case occurs as soon as a process service plan is being executed. The execution environment can request ODERU to provide alternative plans in case of an exception during the execution of process service plan that implements a process model (e.g., when services became unavailable). For this, the plan enacting tool should not only provide the process service plan it tried to execute, but also the current state of execution. That contains information on which services have already been executed, how gateways have been evaluated and what services caused errors during execution. The aim of ODERU in the runtime case is to provide the process execution environment with an alternative solution for the given process *instance*. That is, it tries to patch the current process service plan being stopped in its execution and considers the current state of the world as fixed and not undoable.

3.2 Semantic Annotation of Tasks and Services

In order to be able to automatically compose functionally valid process service plans given a process model, we assume process tasks to be equipped with structured semantic descriptions. Following the SemSOA approach, IOPE of tasks are described in terms of formalized ontological domain knowledge. For the use cases described in this paper, we developed a reference domain ontology called CDM-Core [26], which provides OWL2 descriptions of concepts from the manufacturing domain, in particular for hydraulic metal press maintenance and car exhaust production. The semantic annotations are embedded in the BPMN model by making use of extension elements at the task level.

Similarly, we assume that all services come with semantic annotations of IOPE. For this, the W3C recommendation OWL-S [27] is used, which provides means for not only IOPE annotations, but also for the QoS aspect required for the non-functional optimization. QoS aspects are not predefined in OWL-S, but can be adapted flexibly to the specific use case at hand. Definitions for various QoS aspects are defined in the CDM-Core ontology (or can be defined based on it, in terms of extensions) and could for example represent monetary costs of using a service, operation cycle time of a machine represented by a service or cumulated probability of failure.

3.3 Constraint Optimization Problem Specification

We defined a context-free grammar **COPSE**^{2} to represent constrained optimization problems (COPs) by use of antlr4[1] (cf. Listing 1). The COP specification starts with the definition of the **type** of the COP (linear vs. non-linear, single vs. multi-objective, etc.) and continues with the declaration of the **problem class**.

In this part, the variables, constants and functions are indicated, while in the last segment, any complex function can be defined using operators such as *MAX*, *MIN*, *SUM*, *PRODUCT*, and *IF-ELSE*. The set of **constraints** is then defined with respect to the variables, constants and functions already specified, and the **objective function(s)** is normally constructed by *minimising* one or more functions (or functions combination). In case of a multi-objective, it is possible to have many of them, also in combined form of a *MIN-MAX* COP problem.

Listing 1: *COPSE*^{2} grammar for Constraint Optimization Problems.

```
grammar COPSE2_meta;

problem: 'PROBLEM' type solver problemclass probleminstance output? 'END
       PROBLEM';

type: 'TYPE' Linear Objective 'END TYPE';
Linear: ('linear'|'nonlinear');
Objective: ('single'|'multi');

solver: 'SOLVER' Solver 'END SOLVER';
Solver: ('centralized'|'distributed'|'both');

problemclass: 'CLASS' variables constants? functions? constraints?
       objectivefunction+ 'END CLASS';

variables: 'VARIABLES' (Identifier|ArrayIdentifier)+ 'END VARIABLES';
constants: 'CONSTANTS'  (Identifier|ArrayIdentifier)+ 'END CONSTANTS';
functions: 'FUNCTIONS' function+ 'END FUNCTIONS';
functionSignature: Identifier '(' identifierList ')';
function: functionSignature '=' (expr|ifexpr);

Comparison: '>='|'<='|'=='|'!='|'>'|'<';
Assignment: '=';
expr: '-'? term (('+'|'-') term)*;
term: mterm (('*'|'/'|'^') mterm)*;
dim: Identifier'.length' ;

loop: ('SUM'|'PRODUCT') '(' Identifier ',' (Number|dim) ',' (Number|dim) ','
        expr ')';

mterm: (Identifier|ArrayElem|REAL|'(' expr ')'|('MIN'|'MAX') '{' expr (','
       expr)* '}'|functionSignature|dim|Number|loop);

ifexpr: 'IF' expr Comparison (expr|Number) 'THEN' (expr|ifexpr) 'ELSE' (expr
       |ifexpr) 'END IF';

constraints: 'CONSTRAINTS' constraint+ 'END CONSTRAINTS';
constraint: expr (Comparison|Assignment) (expr|Identifier|Number);

objectivefunction: ('minimize'|'maximize') expr ('->' URI)?;

probleminstance: 'INSTANCE' variabledomains? constantvalues? 'END INSTANCE';

variabledomains: 'DOMAINS' vdomain+ 'END DOMAINS';
constantvalues: 'VALUES' cvalue+ input? 'END VALUES';

input: 'INPUT' inputEntry+ 'END INPUT';
inputEntry: Identifier '<-' '(' Identifier ',' URI ')';
URI: 'http://' ([a-zA-Z0-9/.])+ '#' ([a-zA-Z0-9])+;

vdomain: (Identifier|ArrayIdentifier) ( Number | '[' Number ',' Number ']' |
        '{' Number (',' Number)* '}');
cvalue: (Identifier|ArrayElem) Assignment Number;

output: 'OUTPUT' (valueAssignment|serviceSelection)+ 'END OUTPUT';
valueAssignment: (Identifier|ArrayElem) '->' '(' Identifier ',' URI ')';
serviceSelection: ArrayIdentifier '::' Identifier;

fragment Letter: [a-zA-Z];
fragment ANumber: [0-9];
fragment INF: ('INF'|'-INF');

Number: (('-'? (ANumber+|ANumber* '.' ANumber+) ('*' ('10'|'e') '^' '-'?
       ANumber+)?)|INF);

Identifier: Letter (Letter|ANumber|'_')*;
ArrayIdentifier: Identifier'[]';
ArrayElem: Identifier'['Identifier']';
identifierList: Identifier (',' Identifier)*;

WS: [ \t\r\n]+ -> skip;
```

The **COPSE**^{2} grammar also allows to map back the achieved value to the produced PSP into a semantic concept. In the second part of the constraint optimisation problem definition, the current **problem instance** is indicated: after defining the variables domain

[1] http://www.antlr.org/

Algorithm 1: The pseudocode for the process service plan composition

Input: PM: a semantically annotated BPMN model
Input: S: the set of available services from the repository
parameter: $\mathbf{Sim_{min}}$: minimal similarity value accepted
Output: PSP: the computed process service plan

```
1  % Preparing the data structure
2  forall s ∈ S do
3  |  IOPE_s → IOPE_S;
4  end
5  forall task ∈ PM do
6  |  task → T;
7  end
8  % Find task service candidates
9  forall t ∈ T do
10 |  forall s ∈ S do
11 |  |  if SIM(IOPE_t, IOPE_s) >= Sim_min then
12 |  |  |  s → CANDIDATES_t;
13 |  |  end
14 |  end
15 end
16 % Solve the COP
17 forall t ∈ T do
18 |  forall s ∈ CANDIDATES_t do
19 |  |  forall QoS ∈ T do
20 |  |  |  QoS → Parameters_{s_t};
21 |  |  end
22 |  end
23 end
24 Solutions = COPsolver(Parameters);
25 % Compute a valid data flow
26 forall Solution ∈ Solutions do
27 |  ComposeVariableBindings(Solution) → Plans;
28 end
29 % Compute the Process Service Plans
30 forall Plan ∈ Plans do
31 |  MergePMwithSolution(PM, Plan) → PSPs;
32 |  % Save the computed Process Service Plan into
   |   repository
33 end
34 % Return the first computed Process Service Plan
35 return PSPs[0];
```

and the value of the constants, the mapping of variables values that gives the optimal solution is reported back to semantic concepts used as inputs of the used services.

This approach allows the definition of complex aggregates of QoS and environment variables instead of mere lists of objectives for simple QoS, extending the expressive capability with respect to the non-functional optimisation problem definition. In [28], we showcased how this flexibility can be useful to represent heterogeneous optimisation problems.

3.4 Process Service Planning

The computation of the service plan is presented in Algorithm 1, which uses four helper functions.

The first one is **SIM** ($IOPE_A, IOPE_B$) in line 11, that is used to compute the similarity between two IOPE annotations based on a selected measure. Given the semantic description of a process task ($IOPE_A$) and a service ($IOPE_B$) as input, the adopted measures are the followings:

*Logic-based signature plug-in match of A with B for **I**nputs and **O**utputs*:

$$(\forall i_1 \in I_A, \exists i_2 \in I_B : i_2 \sqsubseteq i_1) \wedge (\forall o_1 \in O_B, \exists o_2 \in O_A : i_2 \sqsubseteq i_1)$$

*Logic specification plug-in match of A with B for **P**recondition and **E**ffects*:

$$KB \models (P_B \Rightarrow P_A) \wedge (E_A \Rightarrow E_B)$$

These matching filters are inspired by the classical plug-in matching of components in software engineering. While a plug-in match is commonly considered near-optimal, we priorize services with semantic descriptions, which are logically equivalent with respect to the requested functionality. A ranking method for logic-based semantic matching filters is proposed in [29]. Alternative approaches to semantic service selection learn the optimal weighted aggregation of different types of non-logic-based and logic-based semantic matching filters [30].

A second helper function is the **COPsolve** (Parameters) used in line 24 for computing a set of Pareto-optimal solutions. This is simply a compiler that transform our COP definition into a running instance of a JaCoP solver[2], using the set of parameters given. A different approach can be anyway reimplemented, if required.

At line 27, the call to **ComposeVariableBindings** (Solution) computes a possible set of variable bindings, which together define the data flow. Bindings are determined by checking the semantic compatibility of the semantic variable types. This ensures a functionally meaningful assignment beyond simple data type compatibility checking. The overall aim of this function is to connect as many service inputs in Solution with outputs of services prior in the execution order determined by the process model definition. Inputs which can not be bound in that way are considered environmental variables (see Listing 2 for examples of both cases). This ensures the direct executability of the computed service plan.

Please note, that the pseudo code leaves out details on handling of gateways and different possible execution paths through the process model for parallel execution and choices. Without loss of generality, the different paths can be considered additional options for generating process service plans, each indicating other gateway decisions and a valid data flow given this decision. ODERU is able to handle parallel (AND), choice (OR) and exclusive (XOR) gateways. While the AND gateway just opens up independent parallel paths and is easy to handle, the XOR and OR gateways result in n and $n!$ possible alternative execution paths, thus widening the problem space significantly. Structurally however, all these options are handled in an analogous way to what explained.

Eventually, **MergePMwithSolution** (PM,Plan) takes care of adding the full metadata section into the original process model to create an executable PSP. This happens at line 31.

[2] http://jacop.osolpro.com/

Listing 2: BPMN snipplet showing the extension for the plan implementation (extract).

```
<?xml version="1.0" encoding="UTF-8"?>
<bpmn:definitions xmlns:bpmn="http://www.omg.org/spec/BPMN/20100524/MODEL"
  xmlns:crema="http://crema.project.eu"
  id="Definitions_1" targetNamespace="http://bpmn.io/schema/bpmn">
  <bpmn:process id="Process_1" isExecutable="true">
   <bpmn:extensionElements>
    <crema:metadata>
     <crema:optimization>
      <crema:formulation><![CDATA[...]]></crema:formulation>
      <crema:results>
       <crema:log><![CDATA[ ...]]></crema:log>
       <crema:dimension name="TotalCost(T,SP)"><crema:value>37</crema:value
            ></crema:dimension>
       <crema:dimension name="TotalTime(T,SP)"><crema:value>22</crema:value
            ></crema:dimension>
       <crema:dimension name="Var1"><crema:value>186.92</crema:value></crema
            :dimension>
      </crema:results>
     </crema:optimization>
     <crema:implementation>
      <crema:service implements="ServiceTask_1yjnl8n" seq="1" origin="
      optimization">
       <crema:marketplaceServiceID>6e0940f0-289f-45ee-b514</crema:
            marketplaceServiceID>
       <crema:owlsDescription>http://.../6e0940f0-289f-45ee-b514.owl</crema:
       owlsDescription>
       <crema:assignments>
         <crema:variable name="Var1" service="6e0940f0-289f-45ee-b514-
       efd533ae9be0" >
           <crema:value>186.92</crema:value>
         <crema:variable>
       </crema:assignments>
       <crema:bindings>
        <crema:binding>
         <crema:origin>
          <crema:variable name="Sp1" service="b5be92ca-a10e-4386-80be-
       ead09a8cb9ce" />
         </crema:origin>
         <crema:target>
          <crema:variable name="Sp1" service="6e0940f0-289f-45ee-b514-
       efd533ae9be0" />
         </crema:target>
        </crema:binding>
        <crema:binding>
         <crema:origin>
          <crema:env />
         </crema:origin>
         <crema:target>
          <crema:variable name="Cu1" service="6e0940f0-289f-45ee-b514-
       efd533ae9be0" />
         </crema:target>
        </crema:binding>
       </crema:bindings>
      </crema:service>
       ...
     </crema:implementation>
    </crema:metadata>
   </bpmn:extensionElements>
   ...
  </bpmn:process>
</bpmn:definitions>
```

3.4.1 Semantic Service selection. The first step of creating a process service plan is to select all possible candidates that are functionally valid for each annotated task of the given process model. For this purpose, we rely on functionally equivalent *exact* or *plug-in* matches [31] limited to direct subclass relationships, in order to have a PSP whose logical properties (in terms of IOPE) are conserved with respect to the given process model.

In the central part of Figure 2, the set of candidates for each task are presented as dashed areas, in which one or multiple services are inserted in descending order of matching. Multiple process service plans can be produced, each differing in the followed path and the variable bindings. From this set of functionally optimal plan candidates the top-k plans are computed which are optimal with respect to the non-functional requirements encoded in the respective COP specification embedded in the process model.

3.4.2 Non-Functional QoS-Based Optimization. The lower part of Fig. 2 shows an example of a result of the non-functional optimization step. Amongst all the possible combinations of services in the candidate pools of the process tasks, the best (or Pareto-optimal in case of multi-objective problem) option is chosen as part of the overall solution. That requires the solving of the COP problem embedded in the extended BPMN[25] description of the process model by minimizing or maximing the specified objective function(s). An extract of a computed valid process service plan is presented in Listing 2, where the results of the COP solution are listed in the section *metadata : optimization : result*. Instead, in the section *metadata : implementation*, the services used for the plan execution are stated together with their input bindings, which ensure optimal execution in terms of constraints and objective functions of the COP. Due to given space limitations, only one service is shown here. Figure 3 presents alternative PSPs, as different options for the process implementation due to the presence of an exclusive gateway.

3.5 ODERU Services Interface

Following the paradigm of XaaS, ODERU uses a RESTful approach for seamless interfacing with any computational platform by implementing the requested services (as from Fig. 1) and using the provided HTTP calls. Every input required is either encoded into the HTTP request address or payload, in the form of a JSON-encoded string. For the list of REST calls and parameters, refer to Table 1. The following gives a brief explanation of ODERU API functionalities:

- "F010: Compose Process Service Plan for Process Model at Design Time" is designed to provide a functionally optimal process service planning for a given process model at design time, before a given deadline. Service composition planning is based on the functional specification of process step of a given process model and available services in terms of their *Inputs, Outputs, Preconditions and Effects (IOPE)*. This means the output is composed of a set of functionally equivalent services for each process step.
- "F020: Optimise Non-Functional Properties of Process Model at Design Time" computes, in near-realtime (i.e: given deadline), a functionally equivalent plan of a composed plan (such as the output of F010, above) satisfying the constraints set and optimising the objective function(s) provided, for a given process instance. The constraints set and the objective function(s), together with the semantically annotated process model, constitute a constrained optimisation (COP) problem.
- "F050: Compose Process Service Plan for Process Instance at Run-time" is designed to provide a functionally optimal process service planning for a given process instance at run-time. This means the output is composed of a set of functionally equivalent services for each process step.
- "F040: Optimise Non-Functional Properties of Process Instance at Run-time" computes, in near-realtime (i.e: given deadline), a functionally equivalent plan of a composed plan (such as the output of F050, below) satisfying the

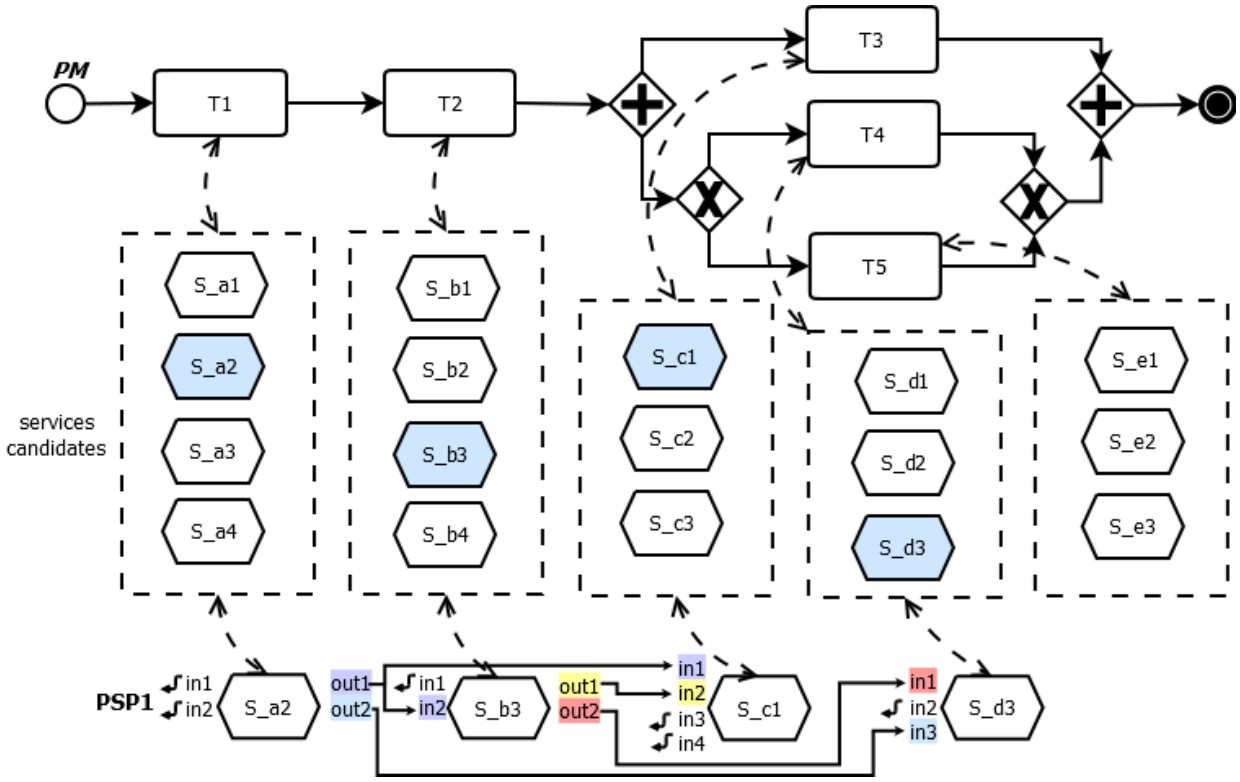

Figure 2: An example of the combined functional and non-functional optimized process service plan: the sequential selection and composition process is shown: for each task all functionally equivalent services are assigned, and then amongst all the possible combinations, the best one is selected based on the result of the COP solving. In case of a request with multiple objectives, one of the Pareto-optimal solutions is returned. Each process service plan is equipped with the relevant variable bindings and optimal service input values for execution.

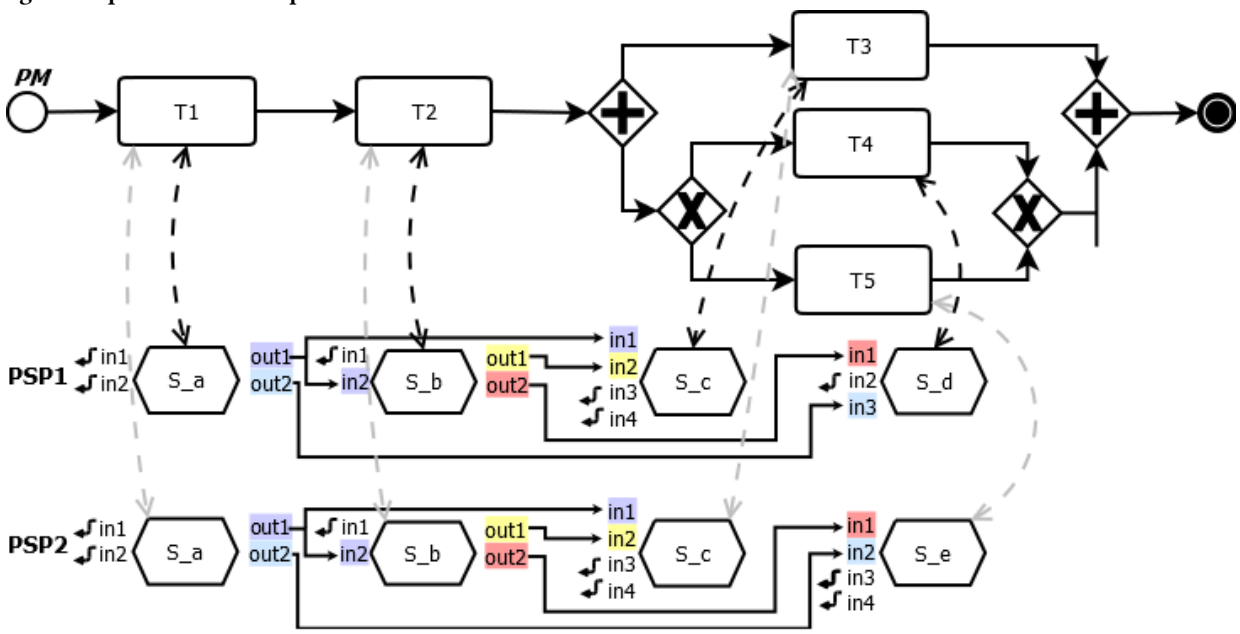

Figure 3: Alternative branches effect: Two possible instances following different paths for the same process models are depicted, as part of the computed solution from ODERU.

Name	URL	JSON payload	JSON return
F010: Compose for PM	PUT /oderu/PM/Compose/	{"ModelID": "PM_1", "deadline": "..."}	"ModelID": "PSP_1"
F020: Optimise for PM	PUT /oderu/PM/Optimise/	{"ModelID": "PM_2", "deadline": "..."}	{"ModelID": "PSP_2", "OptimisationResult": ["http://.../obj_1" : 0.99, "http://.../obj_x" : 0.45], "Assignments": ["http://.../var_1" : 100, "http://.../var_a" : -45.34] }
F050: Compose for PI	PUT /oderu/PI/Compose/	{"ModelID": "PI_3", "deadline": "..."}	"ModelID": "PSP_3"
F040: Optimise for PI	PUT /oderu/PI/Optimise/	{"ModelID": "PI_4", "deadline": "..."}	{"ModelID": "PSP_4", "OptimisationResult": ["http://.../obj_2" : 1.575, "http://.../obj_r" : -12], "Assignments": ["http://.../var_3" : 1.25, "http://.../var_x" : -45.34, "http://.../var_z" : -0.05] }
F030: Approve PSP	PUT /oderu/PSP/Approve/	{"ModelID": "PSP_5", "approval": true}	–
F060: Match Service	GET /oderu/Services/Matching/PM_6/$Task_1$	–	["ServID": "S_1", "similarity": 0.9, "ServID": "S_9", "similarity": 0.87, "ServID": "S_{12}", "similarity": 0.85]
F070: Retrieve PSPs	GET /oderu/PM/Retrieve/PM_7/*timestamp*/	–	[{"ModelID": "PSP_5", "creationTime": "1484056838", "approval": true, "OptimizationResult": ["http://.../obj_1" : 0.99, "http://.../obj_x" : 0.45]}, {"ModelID": "PSP_2", "creationTime": "1484056838", "approval": true, "OptimisationResult": ["http://.../obj_1" : 0.89, "http://.../obj_x" : 0.95]}]
F080: Retrieve PSPs	GET /oderu/PI/Retrieve/PI_8/*timestamp*/	–	[{"ModelID": "PSP_3", "creationTime": "1484056838", "approval": true, "OptimizationResult": ["http://.../obj_a" : 0.08, "http://.../obj_t" : 0.4]}, {"ModelID": "PSP_5", "creationTime": "1484056838", "approval": true, "OptimisationResult": ["http://.../obj_a" : 1.1, "http://.../obj_t" : -0.02]}]
F100: New Service	PUT /oderu/notify/Services/	a valid **ServDTO** see Fragment 3	–
F110: Remove Service	DELETE /oderu/notify/Services/$Serv_1$/	–	–
F120: New Stream chunk	PUT /oderu/notify/Stream/	valid RDF	–

Table 1: The RESTful interfaces of ODERU (all the URL are prefixed by the ODERU deploy address, e.g: http://ODERU.example.org/**). Third and forth columns represent JSON encoded payload and JSON answer payload. All the IDs refers to repositories.**

constraints set and optimising the objective function(s) provided, for a given process instance. The constraints set and the objective function(s), together with the semantically annotated process instance, constitute a constrained optimisation (COP) problem.

- "F030: Approve optimised process service plan" is provided to support the explicit approval (or refusal) of a newly optimised process service plan by the process designer.
- "F060: Find matching services for process step" provides a top-k ranked list of functionally optimal (i.e. semantically most relevant) services that are available to implement a given process step. The semantic relevance computation is based on the hybrid semantic comparison of the semantic IOPE descriptions of process step and available services.
- "F070: Retrieve service plans for process model at design time" is devoted to return already computed optimal process service plans generated after the timestamp indicated as last parameter into the URL (1484056833), with their value of the objective function for a given process model ID. The optional filtering parameter fiapprovalfi allows filtering for the given value of the approval tag (possible values are fitruefi, 'falsefi and 'nullfi).
- "F080: Retrieve service plans for process instance at runtime" is devoted to return previous computed optimal service plans after the timestamp indicated as last parameter into the URL (1484056833), with their value of the objective function for a given process instance ID. The optional filtering parameter fiapprovalfi allows filtering for the given value of the approval tag (possible values are 'truefi, 'falsefi and 'nullfi). For example of its usage please refer to the previous function F070. The number of results returned is limited by the optional filimitfi filtering parameter. This function can be useful to support rapid consideration of already existing and pre-optimised plan.
- "F100: New Semantic Service Notification" is a passive interface used to notify ODERU about the availability of a new service service or the update of an existing one (mainly to be used by the marketplace, for implementing a push approach).
- "F110: Notify Removal of a Semantic Service" is a passive interface used to notify ODERU about the removal of a previously registered service (mainly to be used by the marketplace, for implementing a push approach).
- "F120: Notify availability of new chunk of RDF data stream" is a passive interface to notify ODERU about the availability of new chunk of RDF data stream , in order to allow the internal RDF component to take it into account (to be used by any generic component that wants to communicate stream of data). It is not a streaming interface, due to the limitation of the HTTP protocol, but it emulates a bucket buffered stream.

Two helper functions (not in Table 1) simplify ODERU usage:

- "H510 - Compose Process Service Plan & Optimise Non-Functional Properties for Process Model at Design Time" helps the user by combining (smart pipelining) seamlessly the two functions F010 and F020 for a given process model, in order to create a process service plan optimised both on the functional and non-functional level in a single step.
- "H520 - Compose Process Service Plan & Optimise Non-Functional Properties for Process Instance at Run-Time" supports the user by pipelining, instead, F050 and F040 for a given process instance.

Listing 3: JSON-encoded Service definition, called ServDTO.

```
{
 "serviceID": "Perform_Maintenance_TAS1",
 "isDraft": false,
 "isConcrete": true,
 "serviceName": "http://www.crema-project.eu/UC1.owl#PerformMaintenance",
 "serviceVersion": "1.3",
 "description": "This service implements the maintenance.",
 "serviceOwner": "DFKI",
 "activationDate": 02/08/2016,
 "notifications": "enabled",
 "accessGroups": "all",
 "duration": 1550,
 "QoS": "http://www.crema-project.eu/UC1.owl#Travel_price=0.40Eur/Km,
        http://www.crema-project.eu/UC1.owl#Travel_time=0.01h/Km,
        http://www.crema-project.eu/UC1.owl#Intervention_time=12h,
        http://www.crema-project.eu/UC1.owl#Intervention_price=2000Eur",
 "annotation": {
  "inputs": {
   "semantic": [
    ["http://www.crema-project.eu/UC1.owl#Hydraulic_Press ?i1"],
    ["http://www.crema-project.eu/UC1.owl#Spare_Part ?i2"],
    ["http://www.crema-project.eu/UC1.owl#Customer ?i3"],
    ["http://www.crema-project.eu/UC1.owl#Location ?i4"],
    ["http://www.crema-project.eu/UC1.owl#ScheduledTime ?i5"]
   ],
   "textual": "there is a press ?i1, at client ?i3 located in ?i4, that
         require maintenance at ?i5 using the spare part ?i2"
  },
  "outputs": {
   "semantic": [
    ["http://www.crema-project.eu/UC1.owl#Report ?o1"]
   ],
   "textual": "a report for the intervention ?o1 will be avaialble"
  },
  "preconditions": {
   "semantic": {
    "and": [
     "(http://www.crema-project.eu/UC1.owl#Skill ?s)",
     "(http://www.crema-project.eu/UC1.owl#requires ?i1 ?s)",
     "(http://www.crema-project.eu/UC1.owl#hasSkill TAS1, ?s)",
     "(http://www.crema-project.eu/UC1.owl#SpecialTool ?t)",
     "(http://www.crema-project.eu/UC1.owl#requires ?i1, ?t)",
     "(http://www.crema-project.eu/UC1.owl#has TAS1, ?t)"
    ]
   },
   "textual": "the press ?i1 maintenance requires the skills ?s and the
         special toolset ?t"
  },
  "effects": {
   "semantic": {
    "and": [
     "(http://www.crema-project.eu/UC1.owl#Running ?i1)",
     "(http://www.crema-project.eu/UC1.owl#equippedWith ?i1 ?i2)",
     "(http://www.crema-project.eu/UC1.owl#LocInBasqueCountry ?Ls)",
     "(= ?i4 ?Ls)",
     "(<> ?i5 ?Ls)"
    ]
   },
   "textual": "the press ?i1 mantained with the piece ?i2 is up and running
         smoothly again"
  }
 },
 "linkedConcepts": [
  "http://www.crema-project.eu/UC1.owl#Maintenance",
  "http://www.crema-project.eu/UC1.owl#PreventiveMaintenance",
  "http://www.crema-project.eu/UC1.owl#GoldSupportMaintenance"
 ],
 "serviceSchema": [ "" ],
 "serviceSoftware": "implementation/MaintenanceS$_1$"
 "type" : {
  "inputs": ["xsd:integer", "xsd:integer", "xsd:string", "xsd:string", "xsd:
        dateTime"],
  "outputs": ["xsd:string"]
 }
}
```

3.6 Computational complexity estimation

ODERU is envisioned to work inside a cloud platform for elastic manufacturing in compliance with the SemSOA and XaaS paradigms and can be deployed on any number of node. However, in its current version it does not make use of any parallel and distributed architecture, but every instance works as a an autonomous entity, isolated from any other. In the following, we discuss the computational complexity of the ODERU (Algorithm 1) and identify bottlenecks.

3.6.1 Computing the alternative process service plans. We start our analysis with the computation of functionally optimal process service plans, sketching its dependency to the following dimensions of the given problem:

- T_n is the number of tasks in the considered PM
- S_n is the number of services registered in the repository
- G_n is the number of gateways in the considered PM, divided into:
  - G_{n_P} is the number of parallel gateways (AND)
  - G_{n_E} is the number of exclusive gateways (XOR)
  - G_{n_I} is the number of inclusive gateways (OR)

We have three main parts: *Parsing*, *Matching*, and *Paths computation*, whose complexity is characterized as follows:

$$Complex_{Parsing} := O(T_n + G_n) \tag{1}$$

$$Complex_{Matching} := O(T_n * S_n) \tag{2}$$

$$\begin{aligned} Complex_{Paths} :=& O(G_n) \Rightarrow \\ & O\left(G_{n_P} + \#(G_{n_E}) + \sum_{i=1}^{\#(G_{n_I})} i\right) \Rightarrow \\ & O\left(\#(G_{n_E}) + \sum_{i=1}^{\#(G_{n_I})} i\right) \end{aligned} \tag{3}$$

Putting together the previous equations 1,2, and 3, we obtain an estimation of the complexity as follows:

$$\begin{aligned} Complex :=& Complex_{Parse} + Complex_{Match} + Complex_{Paths} \Rightarrow \\ & O(T_n + G_n) + O(T_n * S_n) + O\left(\#(G_{n_E}) + \sum_{i=1}^{\#(G_{n_I})} i\right) \Rightarrow \\ & O(T_n * S_n) + O\left(\#(G_{n_E}) + \sum_{i=1}^{\#(G_{n_I})} i\right) \end{aligned} \tag{4}$$

under the assumption that the frequency and cardinality of the inclusive and exclusive gateways is comparable, the previous formula can be simplified as follows:

$$Complex := O(T_n * S_n) + O\left(\sum_{i=1}^{\#(G_{n_I})} i\right) \tag{5}$$

To summarize, the implemented algorithm is **linear** in the product $T_n * S_n$ of the number of tasks in the process model received and the number of services registered in the repository (due to the *matching* process in the selection step) and **linear** with respect to the sum of binomial coefficient[3] $\left(\sum_{i=1}^{\#(G_{n_I})} i\right)$ for inclusive gateways cardinalities (due to the *expansion* process in the multiple paths computation step).

In general, it is not possible to determine which of the two aspects has more impact on the complexity, as both depend on the user input. Any set of multiple inclusive gateways with relevant cardinality will dominate, otherwise the product of tasks and available services will determine the computation time for the possible alternative process service plans computation. The presented complexity analysis is valid under the assumption that both the model and the service are correctly annotated in their IOPE profiles. A major issue arises whenever the IOPE is underspecified in tasks and/or services, since the semantic plug-in matching of annotated services with tasks is sensible to the combinatorial explosion of number for the possible services task assignment.

3.6.2 Solving the embedded COP of the process model. A completely different subject is represented by the COP treatment: this is generally independent from the possible alternative process service plan computations. In fact, the user can (and should) design its own objective functions that can refer to any number of tasks and use any number of QoS measures from the services available. Besides, the internal complexity of the optimization library used for the practical COP solving has to be taken into account, since ODERU relies on external libraries for this purpose. In section 5, the two real-world use cases to which ODERU has been applied to clearly show two different types of COP formulations: the first one for press maintenance optimization is concerned with computing the optimal and ranked combinations of maintenance and spare part services for a specific maintenance process instance, while in the second use case for automotive exhaust production, the COP solving by ODERU is done to find the optimal input value configuration of a welding robot service with respect to the overall OEE (Original Equipment Efficiency) of the production line.

4 ILLUSTRATING EXAMPLE AND EVALUATION

As an illustrating example of using ODERU for process optimization, consider a process for the manufacturing of a mechanical metallic part, e.g. a brake disk component (cf. Figure 4). This simple process, after some initial administrative tasks used to retrieve the correct raw material and the production steps, enacts the actual mechanical operation and is concluded by some other administrative jobs that are necessary to associate all the documentation to the produced piece for the delivery to the client, such as the production report and the transportation bill.

In our example, we concentrate only on the process task for the actual manufacturing of the part, as the rest of the actions are only concerned with information management, and the relevant services are normally not the bottlenecks of manufacturing processes. For the implementation of the task 'Mechanical Component', let us assume that there are at least two different services available. The first one, service S_A, wraps a CNC (Computer Numerical Control) equipped machine, is able to directly utilise a CAD/CAM

[3] binomial coefficient : BC := $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

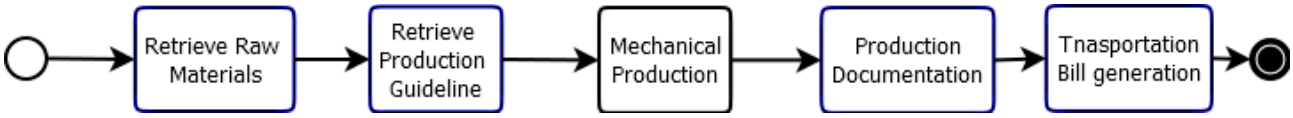

Figure 4: The *Disk Brake* example production model used.

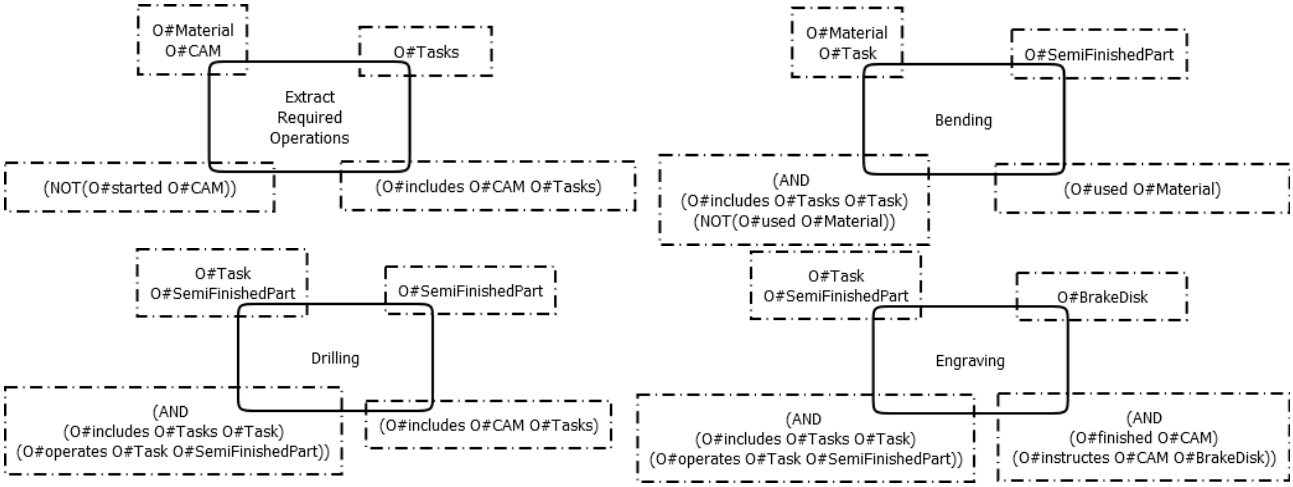

Figure 5: The IOPE semantic annotation for the services S_{B_0} (Extract Required Operations), S_{B_1} (Bending), S_{B_2} (Drilling), S_{B_3} (Engraving). The composition of these services generate an equivalent aggregate of the S_A, from the functional point of view (see Figure 6). In this way, they are interchangable when computing an optimal functional plan implementation for instances.

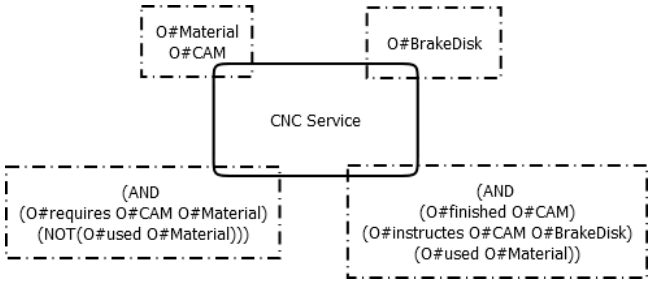

Figure 6: The semantic IOPE annotation of the S_A service based on an ontology O#

(Computer Aided Design/Manufacturing) model for executing a complex set of operations without direct human intervention. For its semantic annotations, we refer to Figure 6, while the QoS parameters of this service are described in Table 2. In addition, the set of services (S_{B_1},S_{B_2},S_{B_3}) implements the three basic operations of bending, drilling and engraving which are required for the mechanical metallic part building. Their semantic IOPE annotations and QoS parameters are shown in Figure 5 and Table 3, respectively. Please note that the service S_A is functionally equivalent with the sequence of services (S_{B_1},S_{B_2},S_{B_3}) but not with respect to the non-functional QoS parameters, as from the extract in Fragment 4.

Let us now formulate the COP for two different instances of the given process model: a first one with an objective function that is dominated by the cost component (i.e: a standard brake disk for economic cars), and a second one where the quality aspect is predominant (i.e: a special part for high range car or a special spare part for tuning purposes). The difference between both instances relies on the two aspects of the process, the CAD/CAM model and the COP formulation. We define the following helper functions (cf. Equations 6,7, and 8):

$$\begin{cases} OF_C(S) = \sum_{i=1}^{S} S[i] * (\phi * Costs[i]) & \phi = 0.1 \\ OF_Q(S) = \sum_{i=1}^{S} S[i] * (\chi * (1 - Quality[i])) & \chi = 5 \\ OF_T(S) = \sum_{i=1}^{S} S[i] * (\psi * Tolerance[i]) & \psi = 10 \end{cases} \quad (6)$$

$$Produced\_Quality(S) = \prod_{i=1}^{S} \begin{cases} 1 & S[i] = 1 \\ Quality[i] & \text{otherwise} \end{cases} \quad (7)$$

$$Produced\_Tolerance(S) = \sum_{i=1}^{S} S[i] * Tolerance[i] \quad (8)$$

The high-range production COP can then be specified as follows:

$$\begin{aligned} & \min_{s \in S} \Big(OF_C(S) + OF_Q(S) + OF_T(S) \Big) \\ s.t. \quad & \sum_{s=1}^{S} Tolerance[s] \leq Limit\_C(= 125) \\ & Produced\_Quality(s) \geq Min\_Q(= 0.50) \\ & Produced\_Tolerance(s) \geq Max\_T(= 3) \end{aligned} \quad (9)$$

The encoding of the COP for the dual instance of *standard* production in the COPSE$_2$ grammar is shown in Listing 4.

	QoS	Value
S_A	Cost	Setup + Execution + CleanUp
S_A	Setup	100
S_A	Execution	22.5
S_A	CleanUp	1.5
S_A	Quality	99.275%
S_A	Tolerance	0.05 mm

Table 2: The QoS measured for the CNC service

	QoS	Value
S_{B_1}	Cost	Setup + Execution
S_{B_1}	Setup	2
S_{B_1}	Execution	5
S_{B_1}	Quality	75%
S_{B_1}	Tolerance	1 mm
S_{B_2}	Cost	Setup + Execution
S_{B_2}	Setup	3
S_{B_2}	Execution	10
S_{B_2}	Quality	79%
S_{B_2}	Tolerance	0.75 mm
S_{B_3}	Cost	Setup + Execution
S_{B_3}	Setup	1
S_{B_3}	Execution	25
S_{B_3}	Quality	85%
S_{B_3}	Tolerance	0.375 mm

Table 3: The QoS measured for the three basic services (bending, drilling, engraving)

Inst	S_A	$S_{B_{1+2+3}}$	Δ **best**	%
I_1	124.005	**46.335**	77.671	62.6%
I_2	99.250	**38.986**	60.264	60.7%

Table 4: Comparison of the objective function values achievable in case where the weights generate a conflict in the assignment for exclusive usage services.

To test the effectiveness of our ODERU solution, we solved the depicted model using the two instances presented in the previous section. As shown in Table 5, it optimizes the two instances for high-range and standard production using two different functionally equivalent implementations, respectively one with a single service S_A and the other with a composed service S_B, resulting from the composition of the three elemetary services S_{B_1}, S_{B_2}, and S_{B_3}.

In this case, the result is clearly indicating a preferred assignment for each instance, but in case of different weights (such as ϕ = 0.8, χ = 0.1, and ψ = 1.0) both instances will be optimized by using the same services (namely, the composition of $\{S_{B_1}, S_{B_2}, S_{B_3}\}$) as reported in Table 4. This is, in case of exclusive usage of resources policy, an issue. However, because the value of the objective function is reported, the user is in condition of deciding which instance to make sub-optimal, maintaining the best global result at the intra-processes level. Despite not being currently fully supported, the development of a specialized module for this is straightforward, given the fact that our current implementation stores all the possible plans (services, sequences, variable bindings and achievable objective value(s)) computed for an instance in a storage facility.

5 INDUSTRIAL USE CASES AND VALIDATION IN PRACTICE

5.1 CREMA Platform and Use Cases in Brief

The ODERU solution for integrated functional and non-functional optimisation of semantic service-based processes has been developed in the European research project CREMA as an integral part of the CREMA platform for cloud manufacturing. Figure 7[4] provides an overview of all components of this platform together with the interactions between them and with the users, and the data exchanges that foster the business logic. For more information on the CREMA platform and its components, we refer to the the

[4]Image taken from the website of the project: http://www.crema-project.eu

Listing 4: COP definition for the PM example in COPSE[2].

```
PROBLEM

 TYPE linear multi END TYPE
 SOLVER both END SOLVER

 CLASS
  VARIABLES
   S
  END VARIABLES

  CONSTANTS
   α β γ Costs[] Quality[] Tolerance[] Limit_C Min_Q Max_T
  END CONSTANTS

  FUNCTIONS
   Objective_Function(S) =  SUM(i,1,S.length,S[i] *  (α * Costs[i] + β * (1
         - Quality{i}) + γ * Tolerance{i}) )
   Produced_Quality(S) = PRODUCT(i,1,S.length, IF S[i] == 1 THEN Quality[i]
        ELSE 1 END IF )
   Produced_Tolerance(S) = SUM(i,1,S.length,S[i] * Tolerance[i])
  END FUNCTIONS

  CONSTRAINTS
   SUM(i,1,S.length, Costs[i]) < Limit_C
   Produced_Quality(S) >=  Min_Q
   Produced_Tolerance(S) < Max_T
  END CONSTRAINTS

  minimize Objective_Function(S) -> http://CREMA/Ont/fake.owl#TaskCosts

 END CLASS

 INSTANCE

  DOMAINS
   S[]{0,1}
  END DOMAINS

  VALUES
    α = 1.0  β = 0.2 γ = 0.1 Limit_C = 125 Min_Q = 0.5 Max_T = 3
   INPUT
    Costs <- (Task_X, http://CREMA/Ont/fake.owl#ServiceCosts)
    Quality <- (Task_X, http://CREMA/Ont/fake.owl#ServiceQuality)
    Tolerance <- (Task_X, http://CREMA/Ont/fake.owl#ServiceTolerance)
  END INPUT
  END VALUES

 END INSTANCE

 OUTPUT
  Produced_Quality(S) -> (Task_X, http://CREMA/Ont/fake.owl#TaskQuality)
  Produced_Tolerance(S) -> (Task_X, http://CREMA/Ont/fake.owl#TaskTolerance)
 END OUTPUT

END PROBLEM
```

Inst	S_A		$S_{B_{1+2+3}}$		**Δ best**	%
I_1	$\alpha * C_A + \beta * (1 - Q_A) + \gamma * T_A$	124.005	$\alpha * C_{B_1+B_2+B_3} + \beta * (1 - \max(Q_{B_1}, Q_{B_2}, Q_{B_3})) + \gamma * T_{B_1+B_2+B_3}$	**46.335**	77.671	62.6%
I_2	$\phi * C_A + \chi * (1 - Q_A) + \psi * T_A$	**12.901**	$\phi * C_{B_1+B_2+B_3} + \chi * (1 - \max(Q_{B_1}, Q_{B_2}, Q_{B_3})) + \psi * T_{B_1+B_2+B_3}$	28.900	15.999	55.4%

Table 5: The comparison of the possible objective function values achievable with the two different alternative implementations for the standard production instance I_1 and the high-range one I_2. The values in *bold* indicate the best solution for each instance (I_1 => $\{S_{B_1}, S_{B_2}, S_{B_3}\}$ and I_2 => S_A), using constant values as from the Listing 4.

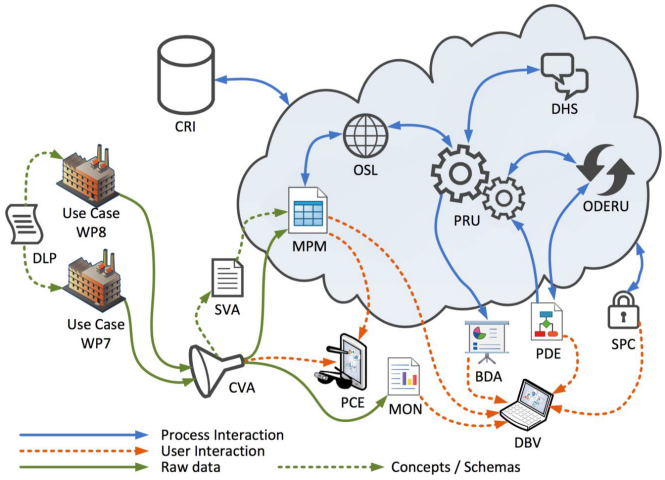

Figure 7: ODERU in context of the CREMA execution architecture.

project website, in particular the "components" page[5] and the appropriate available deliverables. The implemented use case pilots of this platform including ODERU were successfully tested in two different industrial application settings in practice.

5.1.1 Machinery Maintenance Use Case. The first use case in the domain of preventive maintenance was concerned with condition-based optimal maintenance of metal presses with focus on their clutch-brake mechanism without which the presses break down. A special monitoring component continously controls the condition of sensor-equipped press and clutch-brake based on appropriate, expert-defined rules for critical pressure, cooling, friction disc wear, spring fatigue, and braking angle overshootings. In the event of an alarm, the CREMA system obtains the non-functional constraints such as price, warranty and time from the manager and adds the appropriate functional requirements such as needed maintenance skills, tools and spare parts based on the alarm type. The general maintenance process model for the metal press has to be properly instantiated with optimal services for maintenance assistance and spare part provision for these given constraints. In this respect, the process tasks in the model are automatically annotated based on functional requirements with concepts from shared ontology CDM-Core in OWL2, while the non-functional requirements are encoded as constrained optimisation problem and embedded into the extended BPMN specification of the process model instance. ODERU is then used to find the best process service plan for this process model instance based on available and relevant services in the service repository in the cloud. In fact, the objective is to suggest the maintenance manager the optimal combination of assistance teams and of spare parts with minimal costs and time considering potentially competing assignments and respecting hard and soft constraints provided by the client. The manager decides on whether the computed optimal plan gets executed including the tasks of billing and customer feedback at the end othe model instance.

5.1.2 Automotive Use Case. The other process model guides the production of car exhaust filtering systems, by assembling a set of partially-finished parts with the fitting tooling and, eventually, testing the result for product conformity to the client quality requirements. The most relevant process part for ODERU outcome is the selection of the best matching robot cell and operator skills, in order to maximise the OEE (Overall Equipment Effectiveness). This guarantees that the solution performs optimally with respect to measurements for three main aspects of OEE (i.e: availability,performance and quality), where each one of them represents the fraction of normal (good) machine operation given the maximum possible operation time, taking into account the loss caused by several types of problems, usually called the "Six Big Losses" in literature [32, 33].

5.2 Validation and Results

5.2.1 Validation Model. The validation of the CREMA use case pilots based on the V-model, which integrates the waterfall model of the ESA Board of Software Standardisation and Control[34] and the ANSI/IEEE definition of V&V (ANSI/IEEE Std 1012-2004)[35]. The V-model is shown in Fig. 8, where the left part refers to the standard waterfall model of software development, while the right part denotes the processes of verification and validation. The validation of the use case pilots including ODERU in the CREMA project tasks (T7.3, T8.3) refers to the top-level validation process of this model, namely the user acceptance testing with respect to given user-specified functional and business requirements. The integration and system testing was performed during the incremental development process of the CREMA platform components.

The satisfaction of user-specified functional requirements by the use case pilot was tested in respective test scenarios and cases with user-defined criteria of success. The values of business-social performance indicators (BSPI) that were targeted by the industrial user partner for machinery maintenance comply with corresponding user-specified business requirements (cf. Table 6):

- Up to 60% reduction of unscheduled machine downtimes on customers due to a better tracking of critical machine components performance.

[5] http://www.crema-project.eu/components/

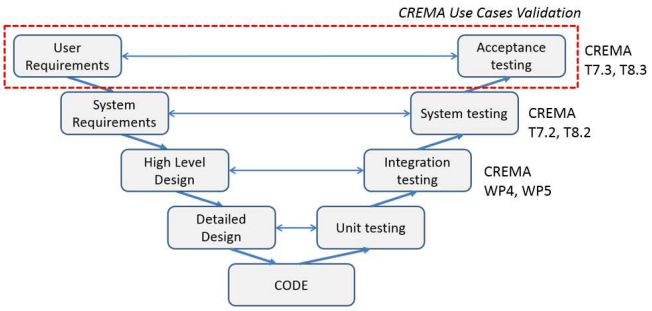

Figure 8: V-model

- 15% reduction of machine downtime for customers due to the increased visibility and on-line monitoring of machines behaviour in customersfi plants.
- Up to 50% reduction of the intervention time on customers since GOIZ will be able to manufacture components in advance that need to be replaced.
- 25% reduction of intervention costs by a better coordination between customers, spare part suppliers (GOIZ), and TAS companies.

The criteria of success have been defined by the user partner as well: The targeted BSPI value has been either fully achieved [pass, P] or partially achieved with 5% tolerance [partial pass, pP], or not at all otherwise [failure, F]. All BSPIs in this industrial use case significantly depend on the capabilities of ODERU to perform its integrated maintenance process optimisation. The BSPIs for the automotive use case together with their achieved status are shown in Table 7 and their success were measured in terms of human expert-based assessments.

5.2.2 Validation Results. The user acceptance testing of the CREMA use case pilot with ODERU for press maintenance process optimisation was quite successful with respect to the satisfaction of both functional and business requirements. In particular, it succeeded in all test cases during its testing at the FAGOR production site in Spain. From 10 user-specified functional requirements with 33 test cases for 12 test scenarios, 10 were completely satisfied (with no partial pass or fail). Notably, the pilot passed 31 (partial pass: 2, fail: 0) test cases completely. In particular, from the 4 user-specified business requirements 3 (1, 0) were completely (partially, not) satisfied. Notably, the corresponding three BSPI values were exceeded with the pilot by quite some extent, while the one BSPI value for the partially satisfied requirement is very close (with less than 1% difference) to the targeted value. The computed BSPI values that are achievable by using the pilot with ODERU for all selected main types of metal press clutch break faults based on the computation of the relevant costs and times over historical data from 2014 until 2017. In addition, the analysis of the users feedback on the pilot gathered from the test users at the FAGOR site was positive; this was reinforced by the impression of other people to whom the pilot has been demonstrated at a CREMA event. The business validation outcomes are shown in Table 6.

The testing of the second CREMA use case pilot that uses ODERU to optimise the welding process of the car exhaust production line at the TENNECO site in the UK turned out to be quite successful as well: 17 of the test cases were marked as Passed, and 4 were marked as Partially Passed, while none of the test cases failed. In particular, out of 5 functional requirements, 3 were completely satisfied and 2 were mostly satisfied; the pilot passed only few test cases partially. Finally, out of 8 BSPIs, 5 were marked as Passed, and 3 were marked as Partially Passed, while none of the corresponding business requirements was not satisfied (cf. Table 7).

For more details including the test environments of both CREMA use case pilots including ODERU, we refer to the respective validation reports which are available from the project website[6].

6 CONCLUSIONS

In this paper, we presented ODERU, a novel solution for the semantic composition of process service plans for annotated process models that are optimal with respect to given functional and non-functional requirements, and showcased the practical benefits of its use in manufacturing. In particular, ODERU computes the set of functionally optimal process service plans in two steps: firstly using the semantically best matching annotated services with respect to the process tasks; and cascade, relying on the computation of top-k fully optimal process service plans based on the solving of an embedded constrained optimisation problem of the process model instance, represented in an extended BPMN format. Finally, the user has to decide on the actual execution of a selected complete and executable optimal process service plan by the process execution environment.

The three main advantages of our pragmatical approach are the integrated combination of functional and non-functional optimization, the complete enactability, thanks to the inclusion also of data flow and optimal variables assignment into the plan, and the fast recomputation capability, enabled by the storage of the plan into extensions of the BPMN formalism.

There are several aspects of potential advancements of ODERU in the future. First, the efficiency and scalability of ODERU may be improved by means of dynamically interleaving both functional and non-functional optimisation. The potentially very large set of computed candidates of functionally optimal process service plans for their subsequent QoS-based optimisation and ranking can result in overall response times that might not be acceptable in practice. Second, the challenge of a proactive event-based triggering of process service plan re-optimization by ODERU could be addressed by integrated means of semantic sensor data stream analysis. Finally, the parallel and distributed computation of process service plans in the cloud based on the model of Hadoop MapReduce[36][7] could speed up the optimisation process of ODERU as well.

The actual version of ODERU is publicly available for download in form of a Docker [37][8] self-contained image under AGPLv3 license at https://oderu.sourceforge.io/.

ACKNOWLEDGMENTS

This work was partially financed by the European Commission H2020 project CREMA (http://www.crema-project.eu), under the

[6] http://www.crema-project.eu/downloads/
[7] http://hadoop.apache.org/
[8] https://www.docker.com/

BSPI	Description	Target	Achieved	Status	relevant
UC1-BS1	CREMA helps in reducing unscheduled machine downtimes.	60%	59,66%	pP	50%
UC1-BS2	CREMA helps in reducing total machine downtimes (scheduled and unscheduled).	15%	58,5%	P	75%
UC1-BS3	CREMA helps in reducing maintenance intervention times.	50%	53,46%	P	100%
UC1-BS4	CREMA helps in reducing maintenance intervention costs.	25%	56,98%	P	75%

Table 6: Business validation results for different BSPIs of the CREMA pilot use case "Machinery Maintenance". The last column evaluates the relevance levels of the ODERU process optimisation for this use case of machine maintenance.

BSPI	Description	Status	relevant
UC2-B0	CREMA increases the speed to allocate production schedule to manufacturing assets.	pP	100%
UC2-B1	CREMA and the location system reduces this time span to introduce new manufacturing assets.	P	–
UC2-B2	CREMA decreases tooling and equipment errors during the tool selection process.	P	25%
UC2-B3	CREMA improves the tooling error resolution time by means of rapid detection and notifications.	P	10%
UC2-B4	CREMA prevents sending untested products to the client.	P	–
UC2-B5	CREMA controls products movement to Area, avoiding that wrong products are sent to the client.	P	–
UC2-B6	CREMA messages the operator with instructions about next steps, helping avoiding human errors.	pP	25%
UC2-B7	CREMA informs operator about problems and avoids time losing with wrong actions.	pP	75%

Table 7: Business validation results for different BSPIs for the pilot of the CREMA use case "Automotive". The last column evaluates the relevance levels of the ODERU process optimisation for this use case of car exhaust welding.

agreement 637066, and the German Federal Ministry of Education and Research (BMBF) in the project INVERSIV.

REFERENCES

[1] S. A. McIlraith, T. C. Son, and H. Zeng, "Semantic web services," *IEEE intelligent systems*, vol. 16, no. 2, pp. 46–53, 2001.

[2] P. Hitzler, M. Krötzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph, "Owl 2 web ontology language primer," *W3C recommendation*, vol. 27, no. 1, p. 123, 2009.

[3] I. Weber, J. Hoffmann, J. Mendling, and J. Nitzsche, "Towards a methodology for semantic business process modeling and configuration," in *Service-Oriented Computing-ICSOC 2007 Workshops*. Springer, 2009, pp. 176–187.

[4] I. Weber, J. Hoffmann, and J. Mendling, "Beyond soundness: on the verification of semantic business process models," *Distributed and Parallel Databases*, vol. 27, no. 3, pp. 271–343, 2010.

[5] M. Hepp, F. Leymann, J. Domingue, A. Wahler, and D. Fensel, "Semantic business process management: A vision towards using semantic web services for business process management," in *IEEE International Conference on e-Business Engineering (ICEBE) 2005*. IEEE, 2005, pp. 535–540.

[6] W. Abramowicz, A. Filipowska, M. Kaczmarek, and T. Kaczmarek, "Semantically enhanced business process modeling notation," in *Semantic Technologies for Business and Information Systems Engineering: Concepts and Applications*. IGI Global, 2012, pp. 259–275.

[7] M. Dimitrov, A. Simov, S. Stein, and M. Konstantinov, "A BPMN based semantic business process modelling environment," in *Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management (SBPM-2007)*, vol. 251, 2007, pp. 1613–0073.

[8] M. Born, J. Hoffmann, T. Kaczmarek, M. Kowalkiewicz, I. Markovic, J. Scicluna, I. Weber, and X. Zhou, "Semantic annotation and composition of business processes with Maestro for BPMN," in *European Semantic Web Conference*. Springer, 2008, pp. 772–776.

[9] D. Karastoyanova, T. van Lessen, F. Leymann, Z. Ma, J. Nitzche, and B. Wetzstein, "Semantic Business Process Management: Applying Ontologies in BPM," in *Handbook of Research on Business Process Modeling*. IGI Global, 2009, pp. 299–317.

[10] M. Klusch, P. Kapahnke, S. Schulte, F. Lecue, and A. Bernstein, "Semantic web service search: a brief survey," *KI-Künstliche Intelligenz*, vol. 30, no. 2, pp. 139–147, 2016.

[11] M. L. Sbodio, "SPARQLent: A SPARQL based intelligent agent performing service matchmaking," in *Semantic Web Services*. Springer, 2012, pp. 83–105.

[12] L.-H. Vu, M. Hauswirth, F. Porto, and K. Aberer, "A search engine for QoS-enabled discovery of semantic web services," *International Journal of Business Process Integration and Management*, vol. 1, no. 4, pp. 244–255, 2006.

[13] T. Pilioura and A. Tsalgatidou, "Unified publication and discovery of semantic web services," *ACM Transactions on the Web (TWEB)*, vol. 3, no. 3, p. 11, 2009.

[14] Y. Zhang, H. Huang, D. Yang, H. Zhang, H.-C. Chao, and Y.-M. Huang, "Bring QoS to P2P-based semantic service discovery for the Universal Network," *Personal and Ubiquitous Computing*, vol. 13, no. 7, pp. 471–477, 2009.

[15] C. Kiefer and A. Bernstein, "The creation and evaluation of iSPARQL strategies for matchmaking," in *European Semantic Web Conference*. Springer, 2008, pp. 463–477.

[16] V. Andrikopoulos and P. Plebani, "Retrieving compatible web services," in *Web Services (ICWS), 2011 IEEE International Conference on*. IEEE, 2011, pp. 179–186.

[17] S. Schulte, U. Lampe, J. Eckert, and R. Steinmetz, "LOG4SWS. KOM: self-adapting semantic web service discovery for SAWSDL," in *Services (SERVICES-1), 2010 6th World Congress on*. IEEE, 2010, pp. 511–518.

[18] M. Klusch and P. Kapahnke, "The iSeM matchmaker: A flexible approach for adaptive hybrid semantic service selection," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 15, pp. 1–14, 2012.

[19] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "PDDL-the planning domain definition language," 1998.

[20] P. Rodriguez-Mier, C. Pedrinaci, M. Lama, and M. Mucientes, "An integrated semantic Web service discovery and composition framework," *IEEE Transactions on Services Computing*, vol. 9, no. 4, pp. 537–550, 2016.

[21] M. Klusch and A. Gerber, "Fast composition planning of owl-s services and application," in *ECOWS'06. 4th European Conference on Web Services, 2006*. IEEE, 2006, pp. 181–190.

[22] A. Strunk, "QoS-aware service composition: A survey," in *2010 IEEE 8th European Conference on Web Services (ECOWS)*. IEEE, 2010, pp. 67–74.

[23] D. Schuller, A. Polyvyanyy, L. García-Bañuelos, and S. Schulte, "Optimization of complex QoS-aware service compositions," in *International Conference on Service-Oriented Computing*. Springer, 2011, pp. 452–466.

[24] G. Zou, Q. Lu, Y. Chen, R. Huang, Y. Xu, and Y. Xiang, "Qos-aware dynamic composition of web services using numerical temporal planning," *IEEE Transactions on Services Computing*, vol. 7, no. 1, pp. 18–31, 2014.

[25] L. Mazzola, P. Kapahnke, P. Waibel, C. Hochreiner, and M. Klusch, "FCE4BPMN: On-demand QoS-based Optimised Process Model Execution in the Cloud," in *Proceedings of the 23rd ICE/IEEE ITMC conference*. IEEE, 2017.

[26] L. Mazzola, P. Kapahnke, M. Vujic, and M. Klusch, "CDM-Core: A Manufacturing Domain Ontology in OWL2 for Production and Maintenance," in *Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - Volume 2: KEOD*, 2016, pp. 136–143.

[27] M. Burstein, J. Hobbs, O. Lassila, D. Mcdermott, S. Mcilraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin *et al.*, "OWL-S: Semantic markup for web services," *W3C Member Submission*, 2004.

[28] L. Mazzola, P. Kapahnke, and M. Klusch, "ODERU: Optimisation of Semantic Service-Based Processes in Manufacturing," in *Proceedings of the 8th International Conference on Knowledge Engineering and Semantic Web (KESW2017), Szczecin, Poland, November 08–10*. Springer, 2017, pp. 337–346.

[29] M. Klusch and P. Kapahnke, "iSem: Approximated reasoning for adaptive hybrid selection of semantic services," in *Extended Semantic Web Conference*. Springer, 2010, pp. 30–44.
[30] M. Klusch, "Overview of the S3 contest: Performance evaluation of semantic service matchmakers," in *Semantic web services*. Springer, 2012, pp. 17–34.
[31] P. Rodriguez-Mier, C. Pedrinaci, M. Lama, and M. Mucientes, "An integrated semantic Web service discovery and composition framework," *IEEE Transactions on Services Computing*, vol. 9, no. 4, pp. 537–550, 2016.
[32] R. Iannone and M. E. Nenni, "Managing oee to optimize factory performance," in *Operations Management*. InTech, 2013.
[33] A. Van Horenbeek, L. Pintelon, A. Bey-Temsamani, and A. Bartic, "Multi-objective optimization of oee (overall equipment effectiveness) regarding production speed and energy consumption."
[34] M. Jones, U. Mortensen, and J. Fairclough, "The esa software engineering standards: Past, present and future," in *Software Engineering Standards Symposium and Forum, 1997. Emerging International Standards. ISESS 97., Third IEEE International*. IEEE, 1997, pp. 119–126.
[35] R. Fujii and D. Wallace, "Software verification and validation," *IEEE Comp. Soc. Press, Los Alamitos*, 1996.
[36] Z. G. Mou and P. Hudak, "An algebraic model for divide-and-conquer and its parallelism," *The Journal of Supercomputing*, vol. 2, no. 3, pp. 257–278, 1988.
[37] J. Turnbull, *The Docker Book: Containerization is the new virtualization*. James Turnbull, 2014.

BIOGRAPHY

Dr. **Luca Mazzola** is research associate at the Hochschule Luzern (HSLU) - Informatik, in the Data Intelligence Research Team, with interest from Big Data to Machine Learning applications, including distributed ledgers and neural network models for data analysis. He holds a MSc. of Engineering in Computer Sciences (Politecnico di Milano, 2004) and a PhD. in Technologies for Human Communication (Università della Svizzera italiana, 2014). He is experienced in secondary usage of information for designing DSS in heterogeneous fields. He was previously employed as researcher at DFKI, applying technologies from semantic Web for manufacturing application. He also lectured Medical Informatics and Computer Sciences classes.
Email: luca.mazzola@hslu.ch
Email: mazzola.luca@gmail.com

Msc. **Patrick Kaphanke** is a computer scientist with extensive experience in the domains of Semantic Web technologies, Semantic Web Services and Deep Learning. After receiving his M.Sc. in Computer Science from the Saarland University in Germany (2006), he worked in the Intelligent Information Systems (I2S) research team headed by PD Dr. Matthias Klusch of the Agents and Simulated Reality (ASR) department at the German Research Center for Artificial Intelligence (DFKI, 2006-2018). Currently, he is working for the German startup company EVANA, which employs AI technologies for intelligent virtual data rooms in the real estate, insurance and legal sectors.
Email: p.kaphanke@evana.de

PD Dr. **Matthias Klusch** is Research Fellow of the German Research Center for Artificial Intelligence (DFKI) and head of the Intelligent Information Systems (I2S) research team of the DFKI department for Agents and Simulated Reality, as well as private docent of computer science at the Saarland University, and chair of the AI section of the German Society for Informatics. He holds a diploma, PhD and habilitation (PD) degree in computer science from the Saarland University. His research focuses on the use of AI, agent and semantic technologies for smart data analysis and service coordination in intelligent information systems. Web: http://www.dfki.de/~klusch
Email: matthias.klusch@dfki.de