Pattern-Based Semantic Composition of Optimal Process Service Plans with ODERU

Luca Mazzola
DFKI, German Research Center for AI
Saarland Informatics Campus D3.2
66123 - Saarbrücken, Germany
Luca.Mazzola@dfki.de

Patrick Kapahnke
DFKI, German Research Center for AI
Saarland Informatics Campus D3.2
66123 - Saarbrücken, Germany
Patrick.Kapahnke@dfki.de

Matthias Klusch
DFKI, German Research Center for AI
Saarland Informatics Campus D3.2
66123 - Saarbrücken, Germany
Matthias.Klusch@dfki.de

ABSTRACT

To keep pace with the needs of the manufacturing industry of the future, companies need to flexibly react to changing demands and be able to manage production capacities in a rapid and efficient way. This requires agile collaboration among supply chain partners in context of Service-Oriented Architectures (SOA). To this end, we propose a novel pragmatic approach for automatically implementing service-based manufacturing processes at design and runtime, called ODERU. Relying on a set of semantic annotations of business process models encoded into an extension of the BPMN 2.0 standard, it combines pattern-based semantic composition of process service plans and optimization of non-functional aspects by means of QoS-based constraint optimization problem (COP) solving. The ODERU tool is part of a platform for cloud-based elastic manufacturing. In this paper we present the foundations of ODERU, showcasing its application to two manufacturing processes with conflicting requirements showing how it solves the problem by leveraging the Everything-as-a-Service (XaaS) approach. Some initial evaluation sketches the expected benefits of such a solution, depicting its usefulness and potentialities.

CCS CONCEPTS

• **Computing methodologies → Planning with abstraction and generalization**; • **Applied computing → Business process modeling**; *Business process management systems*; • **Information systems →** *Semantic web description languages*;

KEYWORDS

QoS based business process optimization, XaaS, SemSOA, BPMN runtime optimization, COP definition for manufacturing.

ACM Reference format:
Luca Mazzola, Patrick Kapahnke, and Matthias Klusch. 2017. Pattern-Based Semantic Composition of Optimal Process Service Plans with ODERU . In *Proceedings of The 19th International Conference on Information Integration and Web-based Applications & Services, Salzburg, Austria, December 4–6, 2017 (iiWAS '17),* 10 pages.
https://doi.org/10.1145/3151759.3151773

1 INTRODUCTION

As every other aspect of the everyday life, also the manufacturing domain is strongly influenced by innovations in ICT. Companies need to flexibly react to changing demands to remain competitive in a dynamic market. This requires multifaceted capabilities, like being able to manage production capacities in a rapid and efficient way and enabling agile collaboration among supply chain partners.

The impact of ICT in this domain is broadly known as Industry 4.0 and ranges from the application of artificial intelligence in robot-assisted production to the usage of IoT devices, always connected and controllable just-in-time.

Along the same line, manufacturing business processes have to be designed and executed in a more dynamic production context, thus creating the need for adaptation and optimization at design time as well as at runtime. As a consequence, the design of process models for business applications need to comprise representations for functional and non-functional requirements beyond what can be specified in traditional BPM (business process modeling) systems, such as semantic representations of product models and manufacturing services as well as KPI requirements and QoS aspects. Moreover, the tools need to be able to provide effective composition of services in the context of SOA (Service-Oriented Architectures) and XaaS (Everything-as-a-Service) systems and reliable model optimization to achieve the best executable service plans for business processes. Eventually, the provided process service plans (PSP) should be designed to support effectively a runtime incremental replanning, in case an included service is temporarily failing or becomes unavailable.

Due to the unavailability of solutions to solve in an integrated way these issues, we developed a novel pragmatic approach called ODERU (**O**ptimization tool for **DE**sign and **RU**ntime). It is able to compose functionally correct plans based on semantic annotations optimizing their non-functional aspects, which are formalized in terms of a Constrained Optimization Problem (COP). The resulting complete service plan (services used, their order, the variable bindings and the environmental variables assignment) is encoded back into specifically developed BPMN 2.0 extensions, bridging the gap between models and executable plans

The paper is organized as follows: in Section II, related work is briefly presented, then we describe the ODERU architecture and algorithm in Section III. Section IV introduces two use cases adopted as application of ODERU. For each of them, a short introduction of the scenario is given, followed by showing the COP definition and eventually describing the design and runtime behavior briefly. The conclusions are given in Section V.

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[1]. 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 [2], 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 [3]. 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 [4] 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 [5] 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. [6] follows the same track with the proposal of BPMO, an ontology, which partly is based on sBPMN, while [7] 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 [8], 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 [9] 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) [10]. 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 [11–13]. 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 [14–16]. 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 [17], which is capable of IOPE selection given formal semantic descriptions in OWL2 and PDDL[2]. 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, [7, 18, 19] 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. [20] provides a survey giving an overview of existing approaches and initiatives in this direction and highlights research questions. In [21], 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 [22], 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

[1]W3C standard; https://www.w3.org/TR/owl2-overview/

[2]Planning Domain Definition Language; http://icaps-conference.org/ipc2008/deterministic/PddlResources.html

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 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: OVERVIEW

Given the problem at hand, we identified three main requirements for ODERU. They are as follows:

(1) it should support integration of functional service selection and composition with non-functional optimization based on flexible measures and objective functions,
(2) as output, it must implement the creation of complete plans, in order to directly enable an execution environment to enact them,
(3) the used format should simplify the re-use and adaptation of the created plans in a dynamic environment, at runtime. This includes the sub-requirement to merge the model information with the service plans details in a single place.

As a result, ODERU performs process service plan composition and QoS-aware plan optimization for given business process models, taking the XaaS (Everything-as-a-Service) perspective and applying the SemSOA methodology. In general, all available resources are assumed to be represented in terms of services and the overall goal of the composition is to assign such services to tasks specified in a given process model.

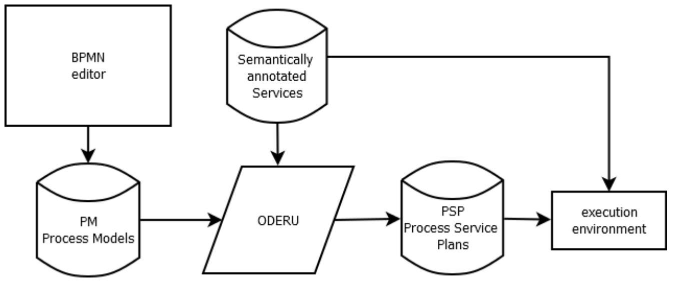

Figure 1: ODERU in context of a fully-fledged BPM and execution architecture.

For an input process model in the semantically enriched extended BPMN format, ODERU computes an executable plan of services implementing the contained tasks including information on the data flow between the services. To provide an optimal solution out of the 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 optimization criteria. Fig. 1 depicts the role of ODERU in 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 aspects for each task. Process models are stored (in a data base, for example in the cloud) and provided to ODERU for performing the process service plan composition and optimization. The resulting process service plans, encoded using another BPMN 2.0 notation extension into the input PM, are stored in a repository and made available for retrieval by an execution environment. Services to be used for planning and later on execution are stored in a semantic service repository.

To achieve this, 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 plug-in services for execution. Analogous to the semantic service descriptions themselves, these annotations are structured in terms of IOPE and refer to domain knowledge in OWL2. Moreover, the BPMN should specify what QoS measures are to be optimized and how they are defined. This is done by specifying a COP at the process model level, whose solutions dictates what services to choose from and what parameter settings to use when calling services. The COP formulation includes information on how to map optimal parameter values to service inputs and service QoS to COP constants. The outputs produced by ODERU are process service plan encoded in the original BPMN itself by making use of 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 as guidance for the execution environment (cf. Fig. 1). The runtime case might appear as soon as a process service plan is executed. The execution environment can query ODERU to provide alternative plans in case of an exception during execution (e.g. service became unavailable). For this, the plan enacting toll should not only provide the process service plan it tried to execute, but also the current state of execution. This includes information on what 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 an alternative solution for the given process *instance*. That is, it tries to patch the existing process service plan and considers the current state of the world as fixed and not undoable.

3.1 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 propose a reference domain ontology called CDM-Core [23], 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 [24] 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.

3.2 Constraint Optimization Problem Definition

We defined an appropriate grammar to represent COPs, based on the requirements of the project use cases, but also taking into account its general re-applicability. The COPSE^2 grammar is defined using antlr4[3] and is presented in Listing 1. It starts by defining the **type** of the COP (linear vs. non-linear, single vs. multi objective, etc.) and continues by declaring the **problem class**. In this part, the variables, constants and functions are indicated. In the last segment, any complex function can be defined, using operators such as *MAX*, *MIN*, *SUM*, *PRODUCT*, and *IF-ELSE*. The **constraints** set 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 you can have many of them, also in comined form of 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;
```

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

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

inputs of the used services. As can be seen, 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 optimization problem definition.

3.3 Process Service Plan

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 10, that is used to compute the similarity between two IOPE annotations based on a selected measure. Given the semantic description of a task ($IOPE_A$) and a service ($IOPE_B$) as input, the adopted measures are the followings:

*Logic-based signature pluging match 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 pluging 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 plugin matching of components in software engineering. While a plugin match is commonly considered near-optimal, we priorize services with semantic descriptions, which are logically equivalent with respect to the to the requested functionality. A possible ranking of logic-based semantic matching filters is proposed for iSeM as shown in [25]. Alternative approaches to semantic service selection learn the optimal weighted aggregation of different types of non-logic-based and logic-based semantic matching filters [26].

A second helper function is the **COPsolve** (Parameters) used in line 23 for computing the set of Pareto-optimal solutions of the COP. This is simple compiler that transform our COP definition into a running instance of a JaCoP solver[4], using the set of parameters given.

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 29.

3.3.1 Services selection. The first step for creating a Process Service Plan is to select all the possible candidates functionally valid for each task. For this step, we rely on functionally equivalent *exact* or on *plug-in* matches [27] limited to direct sub class relationships,

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

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

```xml
<?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>
```

	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 1: 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 2: The QoS measured for the three basic services (bending, drilling, engraving)

in order to have a PSP whose logical properties (in term of IOPE) are conserved with respect to the given PM.

In the central part of the Fig. 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. As can be noticed, every task existing in the process model is considered, as the selection of a valid combination of the task to be actually implemented in the returned process service plan is left for the non-functional optimization, based on the COP solution.

3.3.2 Optimal Services composition. 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 of the candidate pools of the tasks, the best (or Pareto-optimal in case of multiobjective problem) option is chosen as part of the overall solution. This implies solving the COP problem associated to the process model, by minimizing or maximing the objective function(s).

An extract of a possible computed Process Service Plan is presented in Listing 2, where the results of the COP solution are listed in *metadata : optimization : result*. 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 space limitations, only one service is shown here. For details to the BPMN extensions used by ODERU, we refer to [29].

4 AN APPLICATION

To showcase ODERU functionalities, a process for manufacturing a mechanical metallic part was designed (e.g. a brake disk component). Figure 3 depicts it. The 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 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).

For our experiment we concentrate only on the task involved in 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 task 'Mechanical Component',

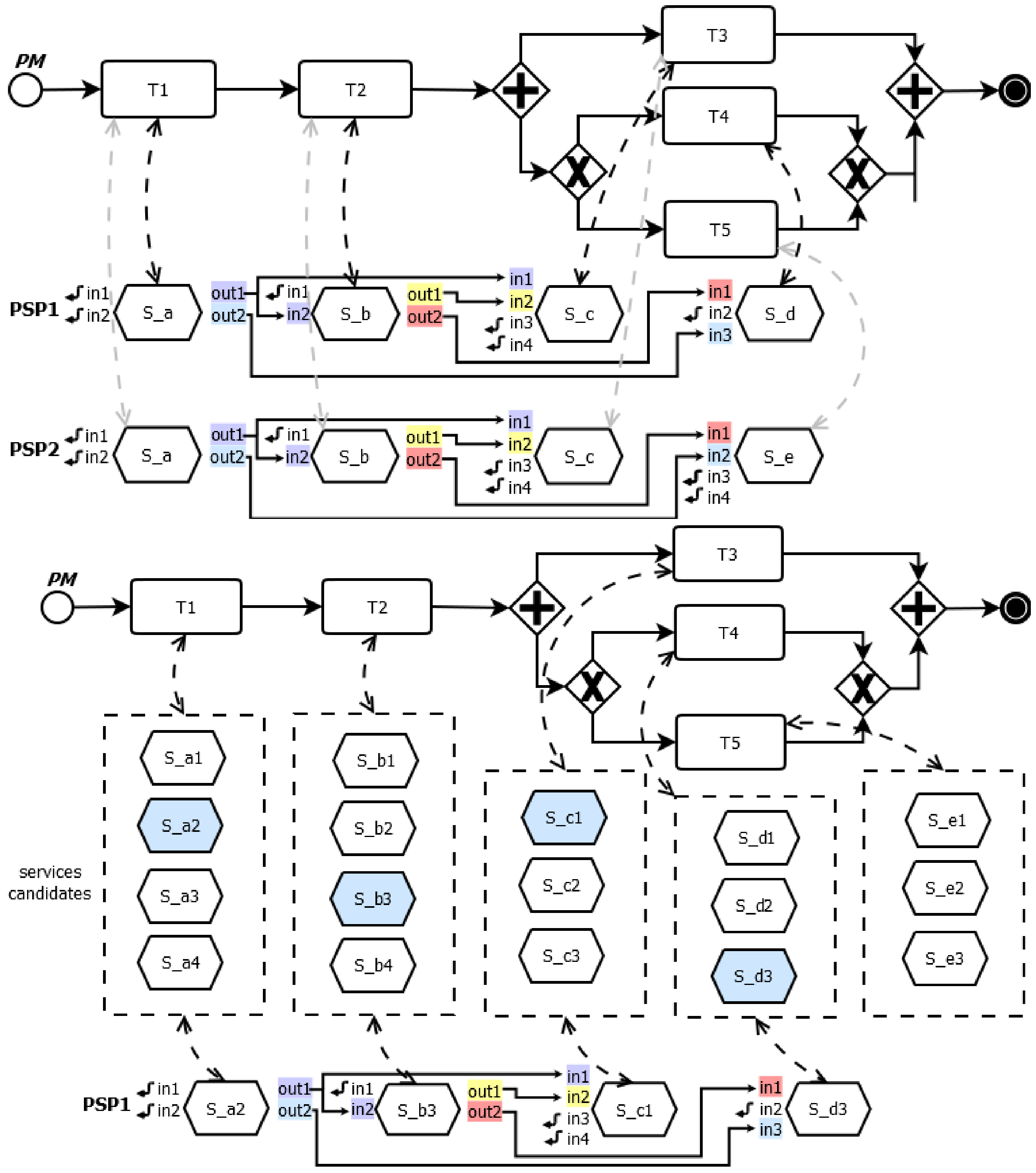

Figure 2: An example of the combined functional and non-functional optimized process service plan. *Upper* two possible instances following different paths for the alternative branch are depicted. *Bottom* The sequential selection and composition process is showed: for each task all the functionally equivalent services are pinpointed then, amongst all the possible combinations, the best one based on the COP formulation is selected and returned for execution. In case of request with multiple objectives, one of the Pareto optimal solution [28] is returned. Each plan is equipped with the relevant variable bindings.

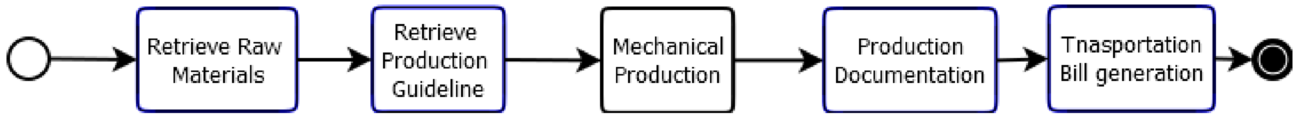

Figure 3: The *Disk Brake* example production model used.

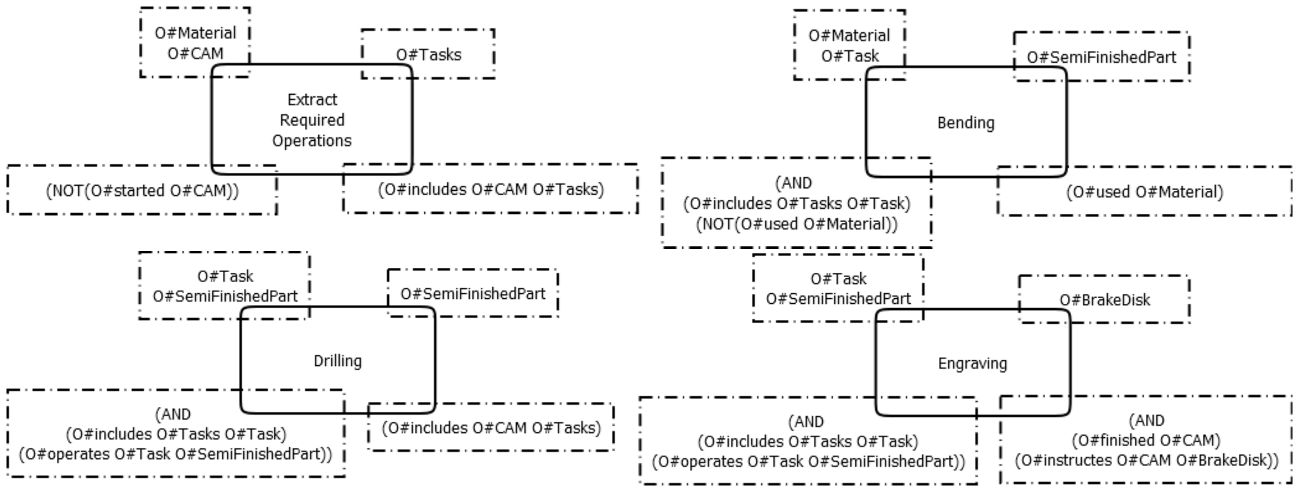
O#Material O#CAM
O#Tasks
Extract Required Operations
(NOT(O#started O#CAM))
(O#includes O#CAM O#Tasks)
O#Material O#Task
O#SemiFinishedPart
Bending
(AND (O#includes O#Tasks O#Task) (NOT(O#used O#Material))
(O#used O#Material)
O#Task O#SemiFinishedPart
O#SemiFinishedPart
Drilling
(AND (O#includes O#Tasks O#Task) (O#operates O#Task O#SemiFinishedPart))
(O#includes O#CAM O#Tasks)
O#Task O#SemiFinishedPart
O#BrakeDisk
Engraving
(AND (O#includes O#Tasks O#Task) (O#operates O#Task O#SemiFinishedPart))
(AND (O#finished O#CAM) (O#instructes O#CAM O#BrakeDisk))

Figure 4: 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 5). In this way, they are interchangable when computing an optimal functional plan implementation for instances.

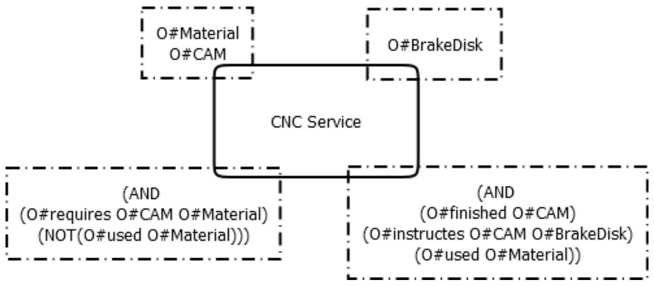
O#Material O#CAM
O#BrakeDisk
CNC Service
(AND (O#requires O#CAM O#Material) (NOT(O#used O#Material)))
(AND (O#finished O#CAM) (O#instructes O#CAM O#BrakeDisk) (O#used O#Material))

Figure 5: The IOPE semantic annotation for the S_A service. Please note that O# indicates an ontology, so every concept and property is fully defined.

we suppose there will be at least two different services available and we will describe them in detail in the next paragraphs.

A first specialised service (S_A) wraps a Computer Numerical Control (CNC) equipped machine, able to directly utilise a Computer Aided Design/Manufacturing (CAD/CAM) for executing a complex set of operations without direct human intervention. For its semantic annotations, please refer to Figure 5. The QoS of this service are described in Table 1. In contrast, a set of services (S_{B_1},S_{B_2},S_{B_3}) implements the three basic operations (bending, drilling, and engraving) that compose the mechanical metallic part building. Their IOPE semantic annotations can be seen in Figure 4. The QoS relevant for these services are described in Table 2. Please note that using the service S_A or the sequence of services (S_{B_1},S_{B_2},S_{B_3}) is equivalent from the functional point of view, as only the non-functional aspect is affected by this choice.

After defining the available services, we describe the COP formulation for two different instance of the given process model: a first one where the objective function 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 the two instances is located inside two aspects of the process: in the CAD/CAM model and in the optimization COP formulation.

While we will not enter into the first aspect, because it is specifically related to the mechanical process and not relevant for the comprehension of the ODERU functionalities, but we present a mathematical formulation of the Contraint Optimisation Problem, starting from some helper functions, as from Equations 1,2, and 3.

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

$$Produced\_Quality(S) = \prod_{i=1}^{S} \begin{cases} 1 & S[i] = 1 \\ Quality[i] & \text{otherwise} \end{cases} \tag{2}$$

$$Produced\_Tolerance(S) = \sum_{i=1}^{S} S[i] * Tolerance[i] \tag{3}$$

Then, the high-range production COP can be represented as in the System 4:

$$
\begin{aligned}
&\min_{s \in S} \Big(OF_C(S) + OF_Q(S) + OF_T(S) \Big) \\
s.t. \quad &\sum_{s=1}^{S} Tolerance[s] \leq Limit\_C(= 125) \\
&Produced\_Quality(s) \geq Min\_Q(= 0.50) \\
&Produced\_Tolerance(s) \geq Max\_T(= 3)
\end{aligned}
\tag{4}
$$

As a showcase and application of the COPSE$_2$ grammar, we present the encoding of the mathematical problem for the dual instance of *standard* production into Fragment 3.

5 INITIAL EVALUATION

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 4, 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}.

Listing 3: The COP definition for the *standard* instance, based on the COPSE$_2$ grammar previously defined.

```
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	124.005	**46.335**	77.671	62.6%
I_2	99.250	**38.986**	60.264	60.7%

Table 3: Comparison of the objective function values achievable in case where the weights generate a conflict in the assignment for exclusive usage services.

In this case, the result is clearly indicating a preferred assignment for each instance, but in case of different weights (such as $\phi = 0.8$, $\chi = 0.1$, and $\psi = 1.0$) 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 3. 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.

6 CONCLUSIONS

In this work we presented our innovative flexible solution to optimal service composition of process models ODERU, which composes functionally correct plans and supports optimization of non-functional aspects, in the form of a Constrained Optimization Problem, using as measures generic QoS and supporting user-defined composed objective functions. To showcase the capabilities of the tool, we applied it for optimising two intances of a mechnical process (disk brake production) in case of service exclusive usage and with conflicting requirements.

Its main advantages in respect of the existing approaches are manifold: the first improvement is the business process formulation: it allows a full integration of functional service selection and composition with non-functional optimization based on user-defined QoS and objective functions arbitrarily complex in the COP. This is achieved through our extensions of the BPMN standard and thanks to the development of a grammar for the optimization part. Secondly, the produced output is directly enactable by an execution environment, being a complete plan. This means that it is equipped with all the relevant information: service assignments, data flow (variable bindings) and optimal variable assignments for initializing the enactment environment. Eventually, the approach used, encoding the computed PSP in an extended BPMN format, allows to maintain in a single place model and plan. This can be useful for a faster and easier reconsideration and/or replanning, if needed.

On the top of the presented use case with two conflicting instances, the ODERU prototype was tested in other contexts, as reported in [30]. These additional tests refer to two initial real-world

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 4: 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 3.

industrial applications in the manufacturing domains of metal press maintenance and automotive exhaust production.

There are still open points we would like to tackle in the future. The most important ones affect (a) the internal ODERU workflow and (b) the usage of data stream information for directing and guide the tool behavior. From the workflow point of view, we are considering how to replace the current fully sequential approach with an interleaving consideration of the functional and non-functional aspects, as we expect better result can be achieved (at least on the efficiency side). We are also considering cases where the possible combinations computed with a pure sequential approach are too numerous to be manageable with a complete approach: in such a case also the effectiveness of the solution can be affected by this change. As for the usage of streamed data from the production shop-floor, we are designing an extension of the ODERU control flow, to allow proactive plan re-optimization triggering.

The current status of the code from the ODERU demonstrator is available for download in form of a Docker self-contained image as AGPLv3 public code at **https://oderu.sourceforge.io/**.

ACKNOWLEDGMENTS

This work was partially financed by the European Commission H2020 RIA project called CREMA (http://www.crema-project.eu), under the agreement 637066. and by 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] 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.
[3] 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.
[4] 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.
[5] 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.
[6] 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.
[7] 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.
[8] 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.
[9] 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.
[10] M. L. Sbodio, "SPARQLent: A SPARQL based intelligent agent performing service matchmaking," in *Semantic Web Services*. Springer, 2012, pp. 83–105.
[11] 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.
[12] 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.
[13] 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.
[14] C. Kiefer and A. Bernstein, "The creation and evaluation of iSPARQL strategies for matchmaking," in *European Semantic Web Conference*. Springer, 2008, pp. 463–477.
[15] V. Andrikopoulos and P. Plebani, "Retrieving compatible web services," in *Web Services (ICWS), 2011 IEEE International Conference on*. IEEE, 2011, pp. 179–186.
[16] 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.
[17] 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.
[18] 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.
[19] 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.
[20] A. Strunk, "QoS-aware service composition: A survey," in *2010 IEEE 8th European Conference on Web Services (ECOWS)*. IEEE, 2010, pp. 67–74.
[21] 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.
[22] 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.
[23] 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.
[24] 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.
[25] 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.
[26] M. Klusch, "Overview of the S3 contest: Performance evaluation of semantic service matchmakers," in *Semantic web services*. Springer, 2012, pp. 17–34.
[27] 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.
[28] C.-L. Hwang and K. Yoon, *Multiple attribute decision making: methods and applications a state-of-the-art survey*. Springer Science & Business Media, 2012, vol. 186.
[29] 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, pp. NN–NN.
[30] L. Mazzola, P. Kapahnke, and M. Klusch, "ODERU: Optimisation of Semantic Service-Based Processes in Manufacturing," in *Knowledge Engineering and Semantic Web - 8th International Conference, KESW 2017, Szczecin, Poland, November 08-10, 2017, Proceedings*, forthcoming, pp. NN–NN.