# Resource Allocation in Business Processes


Giray Havur

Institute for Data, Process and Knowledge Management
Department of Information Systems and Operations Management
Vienna University of Business and Economics

*Supervisors:*
Univ.Prof. Dr. Axel Polleres
PD Dr. Cristina Cabanillas Macias

*Internal reviewer:*
Univ.Prof. Dr. Jan Mendling

*External reviewer:*
O.Univ.Prof. Dr. Thomas Eiter

This cumulative dissertation is submitted for the degree of
*PhD in Economic and Social Sciences*

December 2022

## Abstract

Work is usually divided into discretized and manageable activities within businesses. A series of activities of a similar nature forms a business process that serves an organizational goal, such as producing a good or service. Economic competition often results in businesses trying to operate at the lowest cost and the greatest speed. Therefore, efficient and streamlined execution of business processes by involved resources, such as employees, is important for the overall organization's success. Resource Allocation in Business Processes (RABP) enables effective design-time and run-time selection of the right resources for activity executions and (multi-) objective scheduling of activities, which are vital to improving outcomes for the service and manufacturing industries. The complexity of RABP arises from coordinating explicit and implicit dependencies across a broad set of the process- and resource-related constraints. In this thesis, we: (i) analyze the challenges in Business Processes Management (BPM) to conceptualize RABP; (ii) develop RABP methods to support business process executions; (iii) evaluate the KRR formalisms (i.e., Answer Set Programming and Constraint Programming) with several criteria, such as human readability and maintainability of the RABP problem encoding and computational performance of the automated solvers; and (iv) devise several underpinning methods to improve RAMS (Reliability, Availability, Maintainability, and Safety) of organizational perspective for supporting process executions.

# Contents

## Appendices

# Preface

A growing interest in the corporate ecosystem due to the needs stemming from globalization, integration, standardization, innovation, agility, and operational efficiency has boosted the desire to improve Business Processes (BPs) [1]. A BP consists of a collection of activities in a control flow. When a control flow enables a particular sequence of activities, they are executed by a resource (or multiple resources) to produce a service or product of value to the customer [2–4]. Different methodologies from industrial engineering, operations management, quality management, human capital management, corporate governance, workflow management, and system engineering have all made significant contributions to addressing the improvement of BPs. Business Process Management (BPM) has been initiated to combine and condense these efforts [1]. Typical enhancement targets include cost, time, and failure reductions, which immediately affect businesses' perceived attractiveness in provided goods and services.

As an integral part of BPM, *Resource Allocation in Business Processes* (RABP) enables effective design-time and run-time selection of the right resources for activity executions and (multi-)objective (e.g., optimization of the time, cost, and quality) scheduling of activities. These selections obey temporal and resource-related constraints, which is vital to improving outcomes for the service and manufacturing industries [5]. The complexity of RABP arises from coordinating explicit and implicit dependencies across a broad set of process- and resource-related constraints (i.e., solving potential conflicts on allocating certain resources to activities). Such inter-dependencies relate to control flow in BPs, organizational models, and duration assignments of activities in temporal models.

**Business Process Models:** A clear representation of BPs facilitates pointing out problems related to order of activities, handled data (as a requirement or side-effect of activity execution), and any other aspect involved in process execution. Therefore, it is essential to represent a BP in a format that has the following characteristics [6]:

- *Clear and precise semantics:* The semantics of the BP language is defined formally.
- *Expressive:* The primitives needed to model a BP (e.g., routing constructs, choices, etc.) are supported.
- *Tool-independent:* A language that has mappings to/from different business modeling standards.
- *Verifiable:* There are tools for formally verifying a BP against undesired situations such as deadlocks[1], livelocks[2], improper terminations[3], dead ac-

[1] It is used to describe a process that cannot be completed.

[2] The same activity or sequence of activities are repeatedly executed with no possibility of further continuing towards the end event due to, e.g., a faulty decision node connected to a loop.

[3] The BP execution reaches its end event while there is a live (i.e., still to be executed) activity.

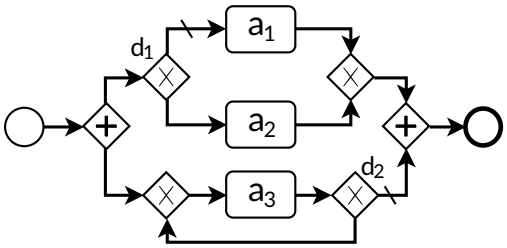



Figure 1: Example BPMN model

tivities[4], and in the context of RABP, resource starvation[5].

Following these characteristics, the Business Process Modeling Notation (BPMN) [7] is used as the *de facto* standard for process modeling. It provides a visual notation that is easy to understand and can be used by both business and technical users. BPMN is widely used for modeling and executing business processes, and is supported by many different tools and platforms. It is also used within our work for mainly representing BPs along with Petri nets [8, 9], which are a special form of graphs or finite automata that can be used to represent processes. BPMN can be mapped to Petri nets [10] (and vice versa), and Petri nets have a formal, mathematical representation with a well-defined syntax and semantics for the definition, validation, and verification of BPs. There are different types of business process modeling methods (besides BPMN), such as BPEL, WS-BPEL, EPC UML Activity diagrams, WF Nets, and YAWL. BPEL [11], or Business Process Execution Language, is a language for specifying and executing business processes. It is based on web services standards, and allows processes to be composed of a set of interconnected web services. BPEL is often used in conjunction with BPMN, with BPMN diagrams being used to model the process and BPEL being used to execute it. WS-BPEL [12], or Web Services Business Process Execution Language, is an extension of BPEL that adds support for more advanced features, such as event handling and compensation. It is widely used for implementing complex, long-running business processes. EPC [13], or Event-driven Process Chain, is a methodology for modeling business processes aiming at covering all the requirements of an information system, and UML Activity diagrams are a way of representing these processes using a standardized graphical notation. WF Nets and YAWL [14] are two different types of workflow modeling languages. WF Nets [15, 16], or Workflow Nets, are a subclass of Petri nets that is specifically designed for modeling business processes. YAWL, or Yet Another Workflow Language, extends WF Nets for specifying and executing complex, flexible business processes requiring advanced synchronization, and cancellation patterns.

[4]At least one activity in the process can never be executed.

[5]The process never reaches the end event due to insufficient resources.

Figure 1 shows an example BPMN model. In this model, the outgoing sequence flows from the parallel gateway (denoted by ⟡) are executed concurrently and synchronized later at the synchronization node (denoted by ⟡ with two incoming sequence flows). At the decision node (denoted by ⟡) $d_1$, only one of the outgoing flows is taken, i.e., either $a_1$ or $a_2$ is executed. Similarly, $a_3$ is executed as many times as necessary (i.e., until the value of the decision variable at $d_2$ is resolved for the sequence flow towards the synchronization node).

RABP can be performed for the entire process before the process execution starts (i.e., at design time) or for a fragment of the process during the process execution continues (i.e., at run time). The process in Figure 1 especially poses a challenge for design-time RABP: needing to know the activities to consider for RABP due to the decision nodes. In this case, some assumptions are made upfront for the decision nodes to derive the input set of activities. However, when these assumptions do not hold at run time, a resource reallocation becomes necessary to repair the schedule. Moreover, concurrent activities (e.g., $a_1$ and $a_3$, or $a_2$ and $a_3$) may require the same resources for their executions, which needs RABP to solve such conflicts while aiming at an optimal schedule.

**Organizational models:** Organizational models contain additional information about available resources within an organization that could contribute to executing activities. Different organizational structures give rise to different organizational models [17]. One of the most widely known models for capturing organizational structures is based on the Role-Based Access Control (RBAC) [18]. An RBAC model describes resources, roles, and a hierarchy of roles that allows the execution of activities based on the roles assigned to them. Roles can be used to model different job positions and scopes of duty within a particular organization. In most cases, the employees of the organization (i.e., its *human resources*) and the infrastructure available (i.e., rooms, machines, etc.) have one or more organizational roles according to their capabilities and characteristics, which allow them to take part in the execution of certain activities.

The organizational meta-model described in [19] explicitly includes concepts like positions, organizational units, capabilities. This meta-model was used to design a language called RAL [20] for defining resource assignment conditions in process models. As presented in [21], RAL can be integrated in existing process modeling notations, such as BPMN, thus enriching the process models with expressive resource assignments that cover various needs described by the creation patterns of the well-known workflow resource patterns [19]. A graphical notation was later designed with the same expressive power as RAL to help the modeler define resource assignments in process models [22].

Moreover, Semantic Web technologies, particularly ontologies, provide appropriate means for representing organizational knowledge in a consistent and coherent format. The Engineering Domain Ontology described in [23] integrates three domains of interest relevant for RABP: BPs, organizational models, and the resource-related constraints arising from regulations and policies [24].

Constraints related to the type of resources to be allocated to process activities add further complexity in RABP: Renewable resources [25] (i.e., resources that are available when not occupied by an activity execution; e.g., employees), non-renewable resources [25] (i.e., resources that have a limited amount of availability from the beginning until the end of a BP execution and that are gradually consumed by the activities competing for these resources; e.g., budget), and partially-renewable resources [25] (i.e., resources whose availability/quantity is renewed at specific periods; e.g., hours per week availability of an employee) require different type of variables to be maintained in a RABP problem encoding.

**Temporal models:** Temporal models comprise the constraints related to the expected duration of the activities, and the tentative deadline for the completion of BP instances. In a typical real-world BPM scenario, activity duration values are estimated by process managers while designing the processes and can be included in the executable BP model as a property of an activity (e.g., with BPMN [7]). On the other hand, temporal constraints can also be estimated by mining *event logs* [26–30] that store evidence of the execution of process instances.

The duration of the activities may be static, resource-dependent, or an aggregate value of the resource-dependent durations when multiple resources are involved in the execution (i.e., collaborative activities in which several employees work together). Furthermore, these values could be defined at design-time or can be estimated dynamically at run-time for the implemented RABP method.

**Optimization in RABP:** Most practical RABP cases require the total length of the schedule called *makespan* to be minimized [31, 32]. Other optimization criteria could also be considered, such as, minimizing the number of late activities and the total delay (i.e., tardiness) of activities [33, 34], balancing the workload of resources [35, 36], and minimizing the overall resource consumption [37, 38]. Including such objectives in RABP methods enables effective resource selection and helps businesses achieve their goals in a shorter time with lower costs and better outcomes.

**Business process management systems:** The BPM lifecycle involves six phases: process identification, discovery, analysis, design, implementation, and monitoring and control [39]. Some of these phases are supported by software systems known as Business Process Management Systems [1] (BPMS). BPMSs aim at providing an integrated set of tools to model, simulate, deploy, enact, monitor, evaluate and continuously optimize BPs. They not only coordinate activities and synchronize data across existing systems but also help streamline activities, triggers, and timelines in a BP.

There are several BPMSs in the market: some are open-source, e.g., Camunda[6], jBPM[7], and Activiti[8]. Others, such as Architecture of Integrated Information Sys-

[6] `https://camunda.com/`
[7] `http://www.jboss.org/jbpm/`
[8] `http://www.activiti.org`

tems (ARIS)[9], Intalio|BPMS[10], and AuraQuantic[11] are commercial. [40, 41] provide insights into strengths and weaknesses of the aforementioned BPMS.

When BPs are run within a BPMS, process execution data (i.e., event logs) is stored. This data contains a series of events for previously run BPs reflecting the start and completion times of activities with the resources involved in their execution. By applying process mining techniques [42–45] on event logs, the extraction of organizational models can be achieved.

**Related literature on RABP:** Resource management in the process- and resource-oriented systems is broken down into three distinct categories in [46]: *resource assignment* (i.e., defining the resources needed for process activities), *resource analysis* (i.e., evaluating process execution with the focus on resources), and *resource allocation* (i.e., assigning concrete resources to a specific task at run time). Resource allocation has been widely used in various domains for addressing everyday problems [47] such as production management [48–58], health care systems [59–67], project management [68–76], maintenance management [77, 78], hotel management [79, 80], reviewers assignment [81–83], education systems [84–87], the military field [88, 89], and sport management [90, 91]. The adoption and representation of (mainly human) resources in the existing process-aware information systems is presented in [92], where the representation of resources and their utilization by the BPs is described in detail (i.e., resource patterns [93]).

We have identified three systematic literature reviews on resource allocation [47, 94, 95] that take into account over 250 papers in total. Let us outline the most remarkable insights and findings of these studies: [94] and [47] show that 82% of the publications deal with only human resources. Even though some of the concepts and patterns also apply to other resources [93], we believe more efforts are needed to describe and utilize non-human resources in the process- and resource-aware information systems. [47] also shows that most of the developed resource allocation methods focus on only 1:1 allocation (i.e., one resource is allocated to one activity; cf. 1:n, n:1, and n:m allocation [47]), which is a limitation for real-world RABP applications.

In [47], the solution methods for RABP fall under four categories: exact (19%), heuristic (32%), metaheuristic (38%), and hybrid (11%) methods, where 71% of the references consider a single optimization objective, and only %29 of the references pursue multi-objective optimization. [95] highlights some of the main RABP approaches in practical contexts, and analyzes them under the lens of key modeling and solution techniques. In doing so, this survey not only reports on a wide plethora of state-of-the-art resource allocation methods but also paves the way for novel techniques by underlining speed-ups in cross-disciplinary artificial intel-

[9] http://www.softwareag.com/en_corporate/platform/aris/business-process-automation.html

[10] http://www.intalio.com/products/process-management/

[11] http://www.auraquantic.com/products/features/business-process-management-bpm/

ligence and operations research models. Moreover, RABP is emphasized as a well-known computationally challenging NP-hard problem, and the reader is reminded that finding solutions to RABP instances under optimization objectives with about 30 activities is still considered a challenging task [96]. As a consequence, identifying novel solution techniques for RABP, and their benchmarks [97, 98] are given prominent importance in RABP-related research [95].

Traditional BPMSs work independently of the area of risk management, which may lead to unfavourable outcomes [99]. BPs should be executed while considering possible negative events and actively managing those risks as part of the process execution [100]. When such events occur, methods (i.e., RABP) that increase runtime resilience against process- and resource-related risks and contingencies for reliable BP executions become necessary [101].

In summary, the state-of-the-art shows that RABP has its particular challenges that are yet to be addressed:

**Challenge 1: Representing a wide variety of resources in RABP.** Resources in a business organization can take many different forms: capital resources (e.g., budget), human resources (e.g., employees in a company), and material resources (e.g., machines in a warehouse) are planned to be used for the execution of BPs in distinct ways [102]. Moreover, legal regulations and compliance rules would affect resource allocation preferences in real-world applications (e.g., separation of duties[12]) [103]. RABP methods in BPM should provide means to incorporate resource characteristics (e.g., roles of resources), the ways the resource are consumed/renewed by activities (e.g., renewable, non-renewable, and partially renewable resources), and resource related constraints that are necessary to cover the needs of BP executions (e.g., break calendars[13], separation of duties, etc.). Existing resource allocation literature in BPM is missing an overarching RABP approach that provides a mechanism to represent a wide variety of resources.

**Challenge 2: Selecting suitable KRR formalisms for implementing RABP.** Declarative Knowledge Representation and Reasoning[14] (KRR) formalisms (i.e., representations that allow for describing the problem and constraints rather than procedures on how to resolve them), as well as respective reasoning methods operating on these representations, are applicable for modeling the RABP method, and for maintaining constraints of specific RABP instances [104]. However, each KRR formalism has a different level of expressiveness and has various execution engines (i.e., solvers, reasoners)

[12]The concept of having more than one employee (or multiple employees that share the same role) required to complete a task.

[13]Break calendar keeps track of the times when renewable resources are unavailable.

[14]Knowledge representation is a study of the symbolic representation of the beliefs, intentions, and judgments for capturing intelligent behavior of an agent. Knowledge Representation and Reasoning (KRR) further focuses on the automated reasoning procedures that can use this knowledge as needed by dedicated reasoning engines to solve complex tasks.

available. RABP implementations in different formalisms need to be compared against each other to find out the most suitable implementation option.

**Challenge 3: Devising a realistic benchmark for testing RABP solutions.** Benchmarking is comparing one's solution for a problem to other solutions via performance metrics. RABP methods are missing a realistic and standardized benchmark due to the complex nature of problem specification that encompasses process-, resource- and time-related requirements in BPM. Such a benchmark would require an RABP problem instance generator that parameterizes resource- and time-related requirements and supports the RABP solutions to run the generated problem instances while logging the performance metrics.

**Challenge 4: Improving reliability of RABP by tackling run-time contingencies.** BPs include stochastic aspects like decision nodes that are used for modeling alternative execution paths. Moreover, the decision nodes may have outgoing branches towards preceding activities in the control flow (i.e., loops). Since the branch-to-be-taken at a decision node depends on the outcome of a previously executed activity, RABP methods need mechanisms for improving the feasibility of allocations. On the other hand, when unexpected execution delays or organizational changes occur at run-time, a reactive RABP mechanism (i.e., for revising pre-established allocations) becomes necessary for reliable executions of BPs.

We formulate the following research questions which we address in this thesis around these four challenges:

RQ.1. *What are the essential requirements for conceptualizing RABP?* (from Challenge 1)

There is no standard description of resource allocation in BPM. Understanding of the essential ideas behind designing and managing BPs and organizational models is crucial for designing a resource allocation method tailored for the needs in BPM [105]. The real-world use cases from the BPM literature [39, 102, 106] help infer those requirements.

RQ.2. *Which are suitable KRR formalisms for implementing RABP?* (from Challenges 2 and 3)

Each formalism (e.g., Answer Set Programming (ASP) [107] and Constraint Programming (CP) [108]) comes with a different set of advantages and disadvantages. The main two criteria are the ease of encoding of the RABP problem (compactness, readability, modularity, and maintainability), and solvers' performance in problem instances. We hypothesize that the KRR methods are now scalable enough to work on real-world problem sizes for RABP.

RQ.3. *How can RABP support process executions to improve their robustness?* (from Challenge 4)

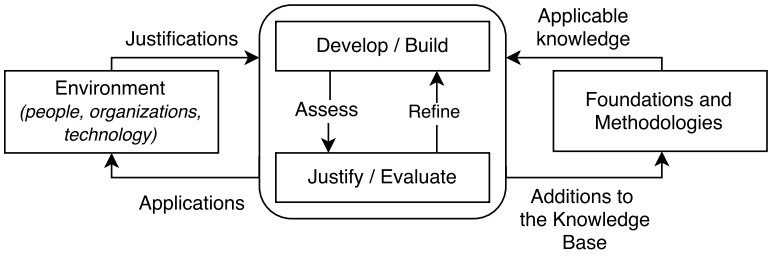



Figure 2: Design science framework (adapted from [111])

BPMSs facilitate the process executions by coordinating all involved activities and resources. Some risks caused by changes in processes, delays in activity executions, and loss of personnel cannot be avoided entirely during process executions [109, 110]. RABP methods should be supported with countermeasures against such adverse events. Using the available BP- and resource-related data, new knowledge can be mined and derived in order to help RABP improve the robustness of process executions.

## Research Method

According to [112] there are two main paradigms that characterize most of the research in information systems: The *behavioral science* paradigm seeks to develop and verify theories that explain or predict human or organizational behavior, and the *design science* paradigm seeks to extend the boundaries of human and organizational capabilities by creating new and innovative artifacts. In other words, behavioral science investigates on developing and justifying theory while design science explores creating innovations and addresses utility. A compact design science framework is shown in Figure 2. For an extended design science research process, we refer to [113] and references therein for details.

As our research is artifact-oriented, it fits into the design science paradigm: (i) we *justify* the relevance of our research by defining the gap in the RABP-related literature and devise a body of knowledge on RABP from the practices in BPM, (ii) we *build* RABP models, methods and supporting tools, (iii) we *assess* these artifacts (e.g., via performance evaluations) and if applicable, test them in real-world scenarios, (iv) we *refine* and further develop the artifacts with respect to the evaluation results, and (v) we *communicate* our research in suitable venues and contribute to the body of knowledge in RABP.

## Selected Papers

This cumulative thesis comprises a collection of publications reflecting the author's contribution to the above-mentioned research questions. Figure 3 presents

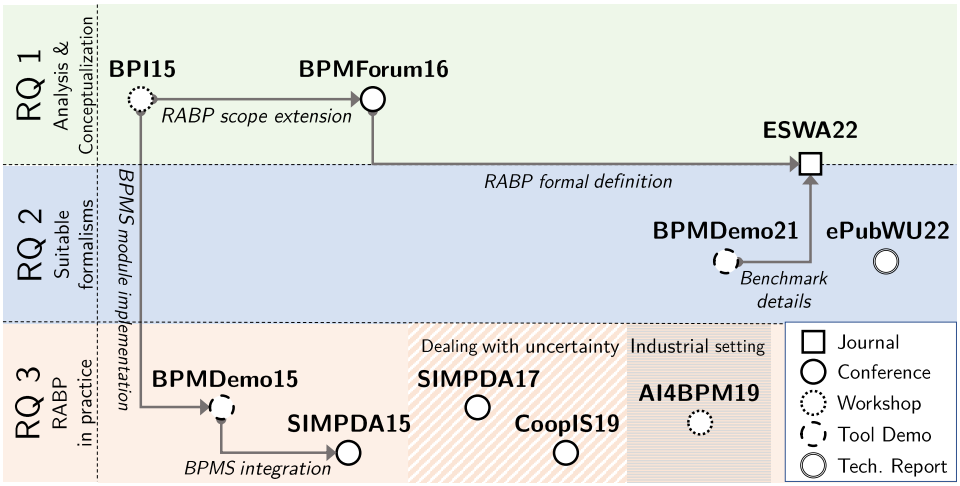



Figure 3: Publications overview

an outline where the venues and the years of the publications are presented. The *relevance to the research questions (1-3)* is represented by the background color: green for the RQ.1, blue for the RQ.2, and orange for the RQ.3. The *type* of a publication is symbolized via the geometry and the outline of the shape, which is presented in the legend: a square represents a journal article; a circle with a solid outline, a conference paper; a circle with a densely dashed outline, a workshop paper; circle with a loosely dashed outline, a tool demo; and a circle with a double outline, a technical report.

**BPI15** [114]: Giray Havur, Cristina Cabanillas, Jan Mendling, and Axel Polleres. Automated resource allocation in business processes with answer set programming. In *Business Process Management Workshops: (BPI 2015)*, Revised Papers, pages 191–203, 2016. This paper details an example scenario that motivates the RABP approach supporting (i) multiple BP instances at once and (ii) BPs that have choice nodes (i.e., when a BP splits into several paths and only one of the paths can be executed) and cycles (i.e., when a previously executed activity in a BP can be returned to during execution). The problem is encoded in incremental ASP for finding a feasible resource allocation with a minimum completion time of all BP instances represented as timed Petri nets.

**BPMForum16** [115] Giray Havur, Cristina Cabanillas, Jan Mendling, and Axel Polleres. Resource allocation with dependencies in business process management systems. In *Business Process Management Forum*, pages 3–19, 2016. This paper describes the conceptualization of RABP under realistic dependencies that affect the outcome of RABP. The high-level concepts such as *resource* and *temporal* requirements of an activity, a complex temporal model (e.g., *specificity levels* of activity durations and *calendar availabil-*

*ity*), and a standardized representation of resource descriptions via the RDF Schema [116] are introduced.

**ESWA22** [117] Giray Havur, Cristina Cabanillas, and Axel Polleres. Benchmarking answer set programming systems for resource allocation in business processes. *Expert Systems with Applications*, 205:117599, 2022. This article is an extended version of [118]: we formalize the RABP problem, provide technical details about the ASP systems benchmark for RABP, and include the benchmark results and their interpretation for one benchmark instance of four state-of-the-art ASP systems in solving RABP.

**ePubWU22** [119] Giray Havur. A comparison of ASP and CP solutions of resource allocation in business processes. Technical report. In *Working papers*, ePub WU, 2022. This technical report includes ASP and Constraint Programming (CP) implementations of RABP. It compares the implementations from two main perspectives: ease of encoding the problem and problem instance solving performance of various solvers of each formalism.

**SIMPDA15** [120] Saimir Bala, Cristina Cabanillas, Alois Haselbock, Giray Havur, Jan Mendling, Axel Polleres, Simon Sperl, and Simon Steyskal. A framework for safety-critical process management in engineering projects. In *SIMPDA 2015, Revised Selected Papers*, volume 244 of LNBIP, pages 1–27, Springer, 2015. This paper delineates the challenges of safety-critical human- and data-centric process management in engineering projects by detailing the components of a framework that allows formalizing human-centric process models, integrating heterogeneous data sources, automating resource allocation, enforcing rules, checking execution of a process against compliance constraints, and adapting process execution when delays occur.

**SIMPDA17** [121] Simon Sperl, Giray Havur, Simon Steyskal, Cristina Cabanillas, Axel Polleres, and Alois Haselbock. Resource utilization prediction in decision-intensive business processes. In *SIMPDA 2017*, volume 2016 of CEUR Workshop Proceedings, pages 128–141, 2017. In this paper, we address the problem of poor utilization of human resources in process executions by describing a mathematical method for quantifying resource utilization with respect to the structural properties of non-deterministic (i.e., decision intensive) processes and the historical executions of these processes.

**CoopIS19** [122] Giray Havur and Cristina Cabanillas. History-aware dynamic process fragmentation for risk-aware resource allocation. In *OTM 2019 Conferences - CoopIS*, volume 11877 of LNCS, pages 533–551, Springer, 2019. In this paper, we introduce a risk-aware process fragmentation method that enhances the feasibility of the resource allocations by dynamically selecting the process fragments (i.e., execution horizons) in run-time concerning the historical process execution data of the process.

**AI4BPM19** [123] Giray Havur, Alois Haselbock, and Cristina Cabanillas. Automated multi-perspective process generation in the manufacturing domain. In *BPM Workshops 2019*, Revised Selected Papers, volume 362 of LNBIP, pages 81–92, Springer, 2019. In this workshop paper, we devise a statistical model from the existing production processes to generate new production processes and resource assignments for new products. This method is validated in an industrial gas turbine production setting and has proven to facilitate the process design efforts for new gas turbines.

## Related papers

BPMDemo15 and BPMDemo21 in Figure 3 are not included in this cumulative thesis. BPMDemo15 is an integration demo of the RABP implementation described in BPI15 into a BPMS engine, and BPMDemo21 is the initial demo paper that paves the way for the journal paper ESWA22.

**BPMDemo15** [124] Saimir Bala, Giray Havur, Simon Sperl, Simon Steyskal, Alois Haselbock, Jan Mendling, and Axel Polleres. Shapeworks: A BPMS extension for complex process management. In *the BPM Demo Track 2016 co-located with BPM 2016*, volume 1789 of CEUR Workshop Proceedings, pages 50–55, 2016. This demo shows a process management framework in complex engineering projects where a BPMS engine (Camunda[15]) is integrated with the RABP implementation in ASP, a semantic model that describes organizational (i.e., resource-related) and safety-critical compliance models, and process mining methods.

**BPMDemo21** [118] Giray Havur, Cristina Cabanillas, and Axel Polleres. BRANCH: An ASP systems benchmark for resource allocation in business processes. In *Proceedings of the Best Dissertation Award, Doctoral Consortium, and Demonstration & Resources Track at BPM 2021*, volume 2973 of CEUR Workshop Proceedings, pages 176–180, 2021. This demo presents the use of the ASP systems benchmark for RABP and its significance to BPM.

### Contribution to research question 1: Analysis and conceptualization

Analysis, conceptualization, and initial implementations of RABP have been presented in BPI15 and BPMForum16. From the BP perspective, BPI15 captures the RABP requirements such as BP execution semantics that allow dealing with cyclic processes. A 1:1 allocation mode (i.e., one resource is allocated to one activity at a time) is implemented using incremental ASP [107, 125]. The devised temporal model not only allows to represent default activity durations but also resource- and role-specific activity durations. A complete formalization of this problem, and its

[15] `https://camunda.org`

a solver-independent ASP encoding in the ASP-Core-2 [126] standard is provided in ESWA22.

In BPMForum16, the representation of resources in BPI15 is extended towards non-renewable and partially-renewable resources with dynamic attributes (i.e., *cumulative resources*). This extension allowed us to encode new allocation modes (besides the 1:1 allocation mode presented in BPI15):

- *n:1 allocation mode*: Multiple resources can be allocated to one activity at a time (e.g., multiple researchers and some research hardware are allocated to one research-related activity).
- *1:n allocation mode*: One resource can be allocated to multiple activities at a time (e.g., a laboratory with limited space is allocated to multiple research-related activities).
- *n:m allocation mode*: Multiple resources can be allocated to multiple activities at a time (e.g., a lab technician and some research hardware is allocated to multiple research-related activities).

Furthermore, the concept of *aggregate durations* (e.g., minimum, maximum, or average of resource-specific activity durations) is implemented to make better duration estimations for n:1 allocations. An aggregate activity duration is computed from many resource- and role-activity durations in the temporal model when an activity is planned to be executed by multiple resources. Finally, an RDFS ontology is formulated for resource descriptions.

**Contribution to research question 2: Suitable formalisms for RABP**

RABP is encoded in the Answer Set Programming (ASP) [127] and Constraint Programming (CP) [128] formalisms: RABP as an incremental answer-set program [107, 125] in BPI15 and BPMForum16, as an ASP-Core-2 [126] compliant answer-set program in ESWA22, and as a constrained optimization problem in the high-level constraint modeling language MiniZinc [129] in ePubWU22.

A configurable ASP systems benchmark for RABP (called BRANCH) is devised and implemented in ESWA22. BRANCH addresses the lack of datasets for benchmarking RABP by providing an RABP problem instance generator. Its easy-to-use interface allows access to the following functionalities: the problem (multi-)instance generator, the ASP system component configurator, the benchmark configurator, the benchmark executor, and the results viewer. An evaluation of four state-of-the-art ASP systems is run using BRANCH, and the detailed performance results (execution time and memory usage) are documented.

A comparison of ASP and CP solutions for RABP is presented in ePubWU22. This technical report summarizes the implementation and performance differences that arise from the formalisms and their corresponding solvers. Our evaluation establishes that ASP provides a compact, easy-to-read, and easy-to-modify problem encoding, whereas CP solvers perform better in solving RABP instances.

**Contribution to research question 3: RABP support in practice**

Several aspects regarding RABP support in practice are studied in SIMPDA15, SIMPDA17, CoopIS19 and AI4BPM19. An RABP module is developed for an automated BPMS solution to address the resource allocation needs of safety-critical engineering projects, such as the deployment of a railway infrastructure, and this module is integrated into the Camunda BPMS in SIMPDA15. The main challenge of RABP in safety-critical domains is the reallocation of resources due to process adaptation needs each time the process monitor observes a discrepancy between the schedule and the BP execution. This problem is solved by first computing the affected activities in the schedule, then applying RABP only on those activities, and finally shifting the starting times of succeeding activities only when necessary. Therefore, the changes made in the schedule become minimal. Furthermore, there are specialized access-control constraints and compliance objectives that are affecting resources. For example, compliance rules such as separation of duties and binding of duties further restrict the workload of human resources and are defined as resource constraints for RABP.

In SIMPDA17, we examine the connection between the control flow of a decision-intensive BP[16] and the anticipated utilization of resources to forecast resource workloads. Our probabilistic forecasting method measures a quantifiable under- and over-utilization risk of resources at a time horizon. This method is tested against a real engineering process, and provides useful insights on identifying the resources expected to be under- and over-utilized, which are difficult to point out before the deployment of the BP.

When dealing with a decision-intensive BP, assumptions on the selection of the branches at the decision nodes are needed to be made in advance to derive a contingent process execution horizon (i.e., the set of activities) for which RABP is performed. Unfortunately, these assumptions may not hold at process run-time, and the discrepancy between the schedule and the execution leads to non-anticipated delays or deadlocks. One way to address this issue is by performing reallocations each time an assumption is overridden. However, if the assumptions are largely inaccurate, this could lead to a sub-optimal use of resources. We introduce in CoopIS19 a novel run-time RABP approach that decides a BP execution horizon at every decision node given a fragmentation threshold. The lower the threshold is, the longer the process fragment becomes, and the risk of facing an unsatisfied assumption in run-time becomes higher. The advantage of this low-threshold/high-risk fragmentation is that the RABP result returns a better resource allocation result as it optimizes the use of resources for a longer execution horizon. A run-time simulation of 40 decision-intensive BPs demonstrates our method's advantages in providing a layer of control on the feasibility of RABP results for decision-intensive processes.

Lastly, in AI4BPM19 we present a multi-perspective (i.e., process and resource-

[16]Decision-intensive BP is densely populated by decision nodes.

related) manufacturing process generation method to support the manual design of manufacturing processes for new products. A statistical model is devised to learn functional, behavioral, and organizational information from the processes for producing products labeled with their features. Afterwards, a production process is generated using this model for a new and labeled product. In the gas turbine manufacturing domain, our method resulted in a significant decrease in time and effort and increased quality in the final process design.

## Acknowledgements

While only my name appears on the cover of this dissertation, this work owes its existence to the time, talents and counsel of a number of people to whom I am indebted.

I owe my deepest gratitude to my supervisor Prof. Dr. Axel Polleres, Head of the Institute for Data, Process and Knowledge Management, whose continuous guidance and support of my academic and career goals have in equal measure pushed and inspired me. His positive criticism and encouragement across these many years of discovery kept me engaged with my research, and his personal generosity helped make this experience deeply meaningful.

I would also like to extend my gratitude to my close collaborator Dr. Cristina Cabanillas, University of Seville, whose enduring commitment to this research (across geographies and institutions) and to offering me thoughtful professional and personal advice sustained and nourished me. I am similarly grateful for the efforts of Prof. Dr. Jan Mendling, Einstein-Professor of Process Science at Humboldt-Universität zu Berlin, whose early engagement with the ideas contained in this work launched this dissertation, and whose mentoring has gone on to enrich my academic horizons. Additionally, I would like to acknowledge the gracious efforts of my committee member, Prof. Dr. Thomas Eiter, TU Wien. Thank you.

My research also benefitted enormously from the insights and good humour of many colleagues who have passed through the corridors of WU and along the several iterations of what was once called the Institute for Information Business – now known as the Institute for Data, Process and Knowledge Management.

This work has also generated a network of colleagues and collaborators whose support, insights and connections have borne many fruits. I will take this opportunity to thank Herwig Schreiner (Siemens AG Österreich), FH-Prof. Mag. Tassilo Pellegrini (Institute for Innovation Systems, FH St. Pölten), and Prof. Dr. Sabrina Kirrane (Institute for Information Systems & New Media, Vienna University of Economics and Business).

Finally, a special thanks to: Prof. Dr. Wil van der Aalst • Prof. Dr. Sinan Açıkgöz • Odai Al Hashmi • Anne-Kathrin Ameling • Ayşe Kıvrak Aykal • Baran Aytaç • Dr. Saimir Bala • Martin Beno • Prof. Dr. Nick Berente • Dr. Stefan Bischof • Dr. Alessio Cecconi • Patrick Comoy • Dr. Richard Comploi-Taupe • Prof. Dr. Ahmet Coşar • Dr. Andreas Falkner • Dr. Marie-France Courriol • Aliz Csapo • Prof. Dr. Claudio Di Ciccio • César Diogo • Katharina

Disselbacher-Kollmann • Djordje Djurica • Saniye Dönmez • Hatice Dönüş • Stefanie Errath • Can Ertaş • Dr. Elif Eryılmaz • Dr. Erwin Filtz • Reinhard Fischer • Roman Franz • Dr. Estefanía García • Dr. Javier David Fernández García • Prof. Dr. Jonas Bulegon Gassen • Prof. Dr. Martin Gebser • Deniz Genç • Dr. Leo Gottlob • Prof. Dr. Thomas Grisold • Dr. Steven Groß • Alexandra Hager • Dr. Alois Haselböck • Mihrican Havur • Magi Hochwarter • Dr. Ayhan İrem • Sami Işıksal • Daniel Janisch • Prabh Jit • Lütfü Kalçık • Candan and Ergun Kancaal • Barışcan Kayıkçı • Jaxson Khan • Dr. Elmar Kiesling • Prof. Dr. Ali Koşar • Can Kurucu • Olivia Labonté • Max Lackner • Mariam Loana Lerch • Ceren Mağden • Dr. Monika Malinova Mandelburger • Paulette and Yüksel Nader • Markus Nagelholz • Dr. Sebastian Neumaier • Christoph Noller • Doğa Okay • Hazal Paftalı • Dr. Josiane Xavier Parreira • Daniela Pico • Hannah Platt • Dr. Sergiu Tcaci Popescu • Jonas Pettersson • Johannes Prescher • Sophie Rogenhofer • Dr. Kate Revoredo • Rebecca Runge • Dr. Miel Vander Sande • Dr. Zeynep Gözen Sarıbatur • Dr. Vadim Savenkov • Prof. Dr. Yücel Saygın • Patrik Schneider • Dr. Ezgi Karakaş Schüller • Dr. Peter Schüller • Maximillian Seunik • Sara Si Ahmed • Gottfried Schenner • Michael Sikic • Dr. Andreas Solti • Simon Sperl • Simon Steyskal • Dr. Elena Theakos • Ioannis Theakos • Liliana Todorovic • Dr. Jürgen Umbrich • Dr. Danilo Valerio • Dr. Svitlana Vakulenko • Julian Vierlinger • Prof. Dr. Ingo Weber • Dr. Alexander Wurl • Doris Wyk • Erdem Yaman • Anton Yeshchenko • Ümit Yiğit • Ayşegül Yüceil • the Families Açıkgöz, Akverdi, Başıbek, Dokuzoğlu, Gökmen, Havur, Kansav, Lackner, Lerch, Platt, and Seunik.

## References


[1] Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A. Reijers. *Fundamentals of Business Process Management, Second Edition*. Springer, 2018.

[2] M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer Verlag, 2012.

[3] Michael Hammer and James Champy. *Reengineering the corporation: a manifesto for business revolution*. HarperBusiness, New York, 1st ed. edition, 1993.

[4] Thomas H. Davenport. *Process innovation: reengineering work through information technology*. Harvard Business School Press, Boston, MA, USA, 1993.

[5] Christoph Schwindt. *Resource allocation in project management*. Springer Science & Business Media, 2006.

[6] Wil MP Van der Aalst. The application of petri nets to workflow management. *Journal of circuits, systems, and computers*, 8(01):21–66, 1998.

[7] OMG. BPMN 2.0. Recommendation, OMG, 2011.

[8] Wil M. P. van der Aalst. The Application of Petri Nets to Workflow Management. *Journal of Circuits, Systems, and Computers*, 8(1):21–66, 1998.

[9] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541 –580, apr 1989.

[10] Niels Lohmann, Eric Verbeek, and Remco Dijkman. Petri Net Transformations for Business

Processes - A Survey. *Transactions on Petri Nets and Other Models of Concurrency II*, 2:46–63, 2009.

[11] OASIS. BPEL 2.0. Recommendation, OASIS, 2007.

[12] Web Services Business Process Execution Language v2.0. Technical report, OASIS, 2007.

[13] August-Wilhelm Scheer, Oliver Thomas, and Otmar Adam. *Process Modeling using Event-Driven Process Chains*, pages 119–145. John Wiley and Sons, Inc., 2005.

[14] Wil M. P. van der Aalst and Arthur H. M. ter Hofstede. YAWL: Yet Another Workflow Language. *Inf. Syst.*, 30(4):245–275, 2005.

[15] Wil M. P. van der Aalst. Verification of workflow nets. In *Application and Theory of Petri Nets 1997*, volume 1248 of *Lecture Notes in Computer Science*, pages 407–426. Springer Berlin / Heidelberg, 1997.

[16] Katalina Grigorova. Process modelling using Petri nets. In *Proceedings of the 4th international conference conference on Computer systems and technologies: e-Learning*, CompSysTech '03, pages 95–100, 2003.

[17] Bryan Horling and Victor Lesser. A Survey of Multi-agent Organizational Paradigms. *Knowledge Engineering Review*, 19(4):281–316, 2004.

[18] Alessandro Colantonio, Roberto Di Pietro, Alberto Ocello, and Nino Vincenzo Verde. A formal framework to elicit roles with business meaning in RBAC systems. In *ACM symposium on Access control models and technologies (SACMAT)*, pages 85–94. ACM, 2009.

[19] Nick Russell, Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and David Edmond. Workflow Resource Patterns: Identification, Representation and Tool Support. In *CAiSE*, pages 216–232, 2005.

[20] Cristina Cabanillas, Manuel Resinas, Adela del Río-Ortega, and Antonio Ruiz-Cortés. Specification and Automated Design-Time Analysis of the Business Process Human Resource Perspective. *Inf. Syst.*, 52:55–82, 2015.

[21] Cristina Cabanillas, Manuel Resinas, and Antonio Ruiz-Cortés. RAL: A High-Level User-Oriented Resource Assignment Language for Business Processes. In *Business Process Management Workshops (BPD'11)*, pages 50–61, 2011.

[22] Cristina Cabanillas, David Knuplesch, Manuel Resinas, Manfred Reichert, Jan Mendling, and Antonio Ruiz-Cortés. RALph: A Graphical Notation for Resource Assignments in Business Processes. In *CAiSE*, volume 9097, pages 53–68. Springer, 2015.

[23] Simon Steyskal, Cristina Cabanillas, Jan Mendling, and Axel Polleres. Engineering Domain Ontology. Project deliverable (d4.1-d4.4), Siemens, 2016.

[24] Simon Steyskal and Axel Polleres. Defining expressive access policies for linked data using the ODRL ontology 2.0. In *SEMANTICS 2014*, pages 20–23, 2014.

[25] Peter Brucker and Sigrid Knust. *Complex scheduling, second edition*. 01 2012.

[26] Wil M. P. van der Aalst. *Process Mining - Data Science in Action, Second Edition*. Springer, 2016.

[27] Eric Rojas, Andres Cifuentes, Andrea Burattin, Jorge Munoz-Gama, Marcos Sepúlveda, and Daniel Capurro. Analysis of emergency room episodes duration through process mining. In *Business Process Management Workshops*, pages 251–263, Cham, 2019. Springer International Publishing.

[28] Edson Ruschel, Eduardo Alves Portela Santos, and Eduardo De Freitas Rocha Loures. Establishment of maintenance inspection intervals: an application of process mining techniques in manufacturing. *J. Intell. Manuf.*, 31(1):53–72, 2020.

[29] Fabrizio M. Maggi, R. P. Jagadeesh Chandra Bose, and Wil M. P. van der Aalst. Efficient discovery of understandable declarative process models from event logs. In *Advanced Information Systems Engineering*, pages 270–285, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[30] Joyce Nakatumba-Nabende and Wil Aalst. Analyzing resource behavior using process mining. volume 43, pages 69–80, 09 2009.

[31] Weidong Zhao, Qingfeng Zeng, Guangjian Zheng, and Liu Yang. The resource allocation model for multi-process instances based on particle swarm optimization. *Inf. Syst. Frontiers*, 19(5):1057–1066, 2017.

[32] Jiajie Xu, Chengfei Liu, Xiaohui Zhao, and Zhiming Ding. Incorporating structural improvement into resource allocation for business process execution planning. *Concurr. Comput. Pract. Exp.*, 25(3):427–442, 2013.

[33] Malgorzata Sterna. A survey of scheduling problems with late work criteria. *Omega*, 39(2):120–129, 2011.

[34] J. N. Hooker. Planning and scheduling to minimize tardiness. In Peter van Beek, editor, *Principles and Practice of Constraint Programming - CP 2005*, pages 314–327, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[35] Sunita Singhal and Jemin Patel. Load balancing scheduling algorithm for concurrent workflow. *Comput. Informatics*, 37(2):311–326, 2018.

[36] Jeremy Decerle, Olivier Grunder, Amir Hajjam El Hassani, and Oussama Barakat. Impact analysis of workload balancing on the home health care routing and scheduling problem. In *2017 4th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 0096–0101, 2017.

[37] Yain-Whar Si, Veng-Ian Chan, Marlon Dumas, and Defu Zhang. A petri nets based generic genetic algorithm framework for resource optimization in business processes. *Simul. Model. Pract. Theory*, 86:72–101, 2018.

[38] Farah Bellaaj, Mohamed Sellami, Sami Bhiri, and Zakaria Maamar. Obstacle-aware resource allocation in business processes. In *Business Information Systems - 20th International Conference, BIS 2017, Poznan, Poland, June 28-30, 2017, Proceedings*, volume 288 of *Lecture Notes in Business Information Processing*, pages 207–219. Springer, 2017.

[39] Marlon Dumas, Marcello La Rosa, Jan Mendling, Hajo A Reijers, et al. *Fundamentals of Business Process Management*, volume 1. Springer, 2013.

[40] Fraunhofer IESE. Studie - BPM SUITES 2013. `http://www.iese.fraunhofer.de/`, 2013.

[41] Giray Havur, Simon Steyskal, Alex Wurl, Cristina Cabanillas, Jan Mendling, and Polleres Axel. State-of-the-art report on existing models for processes, resources, constraints and security and their underlying formalisms. SHAPE project deliverable (D2.1), Vienna University of Economics and Business, 2015.

[42] Maria Leitner, Anne Baumgrass, Sigrid Schefer-Wenzl, Stefanie Rinderle-Ma, and Mark Strembeck. A case study on the suitability of process mining to produce current-state RBAC models. In *Business Process Management Workshops - BPM 2012 International Workshops, Tallinn, Estonia, September 3, 2012. Revised Papers*, volume 132 of *Lecture Notes in Business*

*Information Processing*, pages 719–724. Springer, 2012.

[43] Anne Baumgrass and Mark Strembeck. Bridging the gap between role mining and role engineering via migration guides. *Inf. Secur. Tech. Rep.*, 17(4):148–172, 2013.

[44] Weidong Zhao and Xudong Zhao. Process mining from the organizational perspective. In *Foundations of intelligent systems*, pages 701–708. Springer, 2014.

[45] Stefan Schönig, Cristina Cabanillas, Stefan Jablonski, and Jan Mendling. Mining the Organisational Perspective in Agile Business Processes. In *BPMDS*, pages 37–52, 2015.

[46] Cristina Cabanillas. Process- and resource-aware information systems. *EMISA Forum*, 37(1):40–41, 2017.

[47] Sana Bouajaja and Najoua Dridi. A survey on human resource allocation problem and its applications. *Oper. Res.*, 17(2):339–369, 2017.

[48] Xavier Boucher, Eric Bonjour, and Bernard Grabot. Formalisation and use of competencies for industrial performance optimisation: A survey. *Computers in industry*, 58(2):98–117, 2007.

[49] Leonardo Borba and Marcus Ritt. A heuristic and a branch-and-bound algorithm for the assembly line worker assignment and balancing problem. *Comput. Oper. Res.*, 45:87–96, 2014.

[50] Robert L. Burdett and Erhan Kozan. The assignment of individual renewable resources in scheduling. *Asia Pac. J. Oper. Res.*, 21(3):355–378, 2004.

[51] Albert Corominas, Rafael Pastor, and Ericka Rodríguez. Rotational allocation of tasks to multifunctional workers in a service industry. *International Journal of Production Economics*, 103(1):3–9, 2006.

[52] A Costa, FA Cappadonna, and S Fichera. A hybrid genetic algorithm for job sequencing and worker allocation in parallel unrelated machines with sequence-dependent setup times. *The International Journal of Advanced Manufacturing Technology*, 69(9):2799–2817, 2013.

[53] Horst A Eiselt and Vladimir Marianov. Employee positioning and workload allocation. *Computers & operations research*, 35(2):513–524, 2008.

[54] Cristóbal Miralles, José P García-Sabater, Carlos Andrés, and Manuel Cardós. Branch and bound procedures for solving the assembly line worker assignment and balancing problem: Application to sheltered work centres for disabled. *Discrete Applied Mathematics*, 156(3):352–367, 2008.

[55] Özcan Mutlu, Olcay Polat, and Aliye Ayca Supciller. An iterative genetic algorithm for the assembly line worker assignment and balancing problem of type-ii. *Computers & Operations Research*, 40(1):418–426, 2013.

[56] Syed Mithun Ali. U-shaped assembly line balancing with temporary workers. *International Journal of Industrial Engineering*, 21(6):134–146, 2015.

[57] Sicong Tan, Wei Weng, and Shigeru Fujimura. Scheduling of worker allocation in the manual labor environment with genetic algorithm. In *Proceedings of the international multiconference of engineers and computer scientists*, volume 1, 2009.

[58] Mariona Vila and Jordi Pereira. A branch-and-bound algorithm for assembly line worker assignment and balancing problems. *Computers & Operations Research*, 44:105–114, 2014.

[59] Huabo Zhu, Jiafu Tang, and Jun Gong. Nurses staffing and allocation in multi-stage queueing network with i2 patients' routing for outpatient department. *Journal of Applied Sciences*,

13(15):2884–2890, 2013.

[60] Giuliana Carello and Ettore Lanzarone. A cardinality-constrained robust model for the assignment problem in home care services. *European Journal of Operational Research*, 236(2):748–762, 2014.

[61] Cicero Ferreira Fernandes Costa Filho, Dayse Aparecida Rivera Rocha, Marly Guimarães Fernandes Costa, and Wagner Coelho de Albuquerque Pereira. Using constraint satisfaction problem approach to solve human resource allocation problems in cooperative health services. *Expert Systems with Applications*, 39(1):385–394, 2012.

[62] Ettore Lanzarone and Andrea Matta. Robust nurse-to-patient assignment in home care services to minimize overtimes under continuity of care. *Operations Research for Health Care*, 3(2):48–58, 2014.

[63] Heshani C Rathnayake and Shalinda Adikari. Swarm intelligence for resource allocation of emergency situations in hospitals. In *2013 8th International Conference on Computer Science & Education*, pages 446–451. IEEE, 2013.

[64] Qian Zheng, Jie Shen, Ze-qing Liu, Kai Fang, and Wei Xiang. Resource allocation simulation on operating rooms of hospital. In *2011 IEEE 18th International Conference on Industrial Engineering and Engineering Management*, pages 1744–1748. IEEE, 2011.

[65] Pedro M Castro and Inês Marques. Operating room scheduling with generalized disjunctive programming. *Computers & Operations Research*, 64:262–273, 2015.

[66] Thiago AO Silva, Mauricio C de Souza, Rodney R Saldanha, and Edmund K Burke. Surgical scheduling with simultaneous employment of specialised human resources. *European Journal of Operational Research*, 245(3):719–730, 2015.

[67] Atle Riise, Carlo Mannino, and Edmund K Burke. Modelling and solving generalised operational surgery scheduling problems. *Computers & Operations Research*, 66:1–11, 2016.

[68] V Shahhosseini and MH Sebt. Competency-based selection and assignment of human resources to construction projects. *Scientia Iranica*, 18(2):163–180, 2011.

[69] Christos Kyriklidis, Vassilios Vassiliadis, Konstantinos Kirytopoulos, and Georgios Dounias. Hybrid nature-inspired intelligence for the resource leveling problem. *Operational Research*, 14(3):387–407, 2014.

[70] Dongwon Kang, Jinhwan Jung, and Doo-Hwan Bae. Constraint-based human resource allocation in software projects. *Software: Practice and Experience*, 41(5):551–577, 2011.

[71] Martina Huemann, Anne Keegan, and J Rodney Turner. Human resource management in the project-oriented company: A review. *International journal of project management*, 25(3):315–323, 2007.

[72] MHA Hendriks, B Voeten, and L Kroep. Human resource allocation in a multi-project r&d environment: resource capacity allocation and project portfolio planning in practice. *International journal of project management*, 17(3):181–188, 1999.

[73] Antonella Certa, Mario Enea, Giacomo Galante, and Concetta Manuela La Fata. Multi-objective human resources allocation in r&d projects planning. *International Journal of Production Research*, 47(13):3503–3523, 2009.

[74] Ming-Fung Francis Siu, Ming Lu, and Simaan AbouRizk. Methodology for crew-job allocation optimization in project and workface scheduling. In *ASCE*, pages 652–659, 2015.

[75] Wail Menesi, Mohamed Abdel-Monem, Tarek Hegazy, and Zinab Abuwarda. Multi-objective

schedule optimization using constraint programming. In *ICSC15*, 2015.

[76] Arno Sprecher and Andreas Drexl. Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm. *European Journal of Operational Research*, 107(2):431 – 450, 1998.

[77] Bertrand Estellon, Frédéric Gardi, and Karim Nouioua. High-performance local search for task scheduling with human resource allocation. In *International Workshop on Engineering Stochastic Local Search Algorithms*, pages 1–15. Springer, 2009.

[78] MEZIANE Bennour, S Addouche, and ABDERRAHMAN El Mhamedi. Rcpsp sous contraintes de compétences dans un service de maintenance. *European Journal of Information Systems*, 46(8):877–907, 2012.

[79] Kayoko Murakami, Mitsuo Gen, Seren Ozmehmet Tasan, and Takashi Oyabu. A solution of human resource allocation problem in a case of hotel management. In *The 40th International Conference on Computers & Indutrial Engineering*, pages 1–6. IEEE, 2010.

[80] Kayoko Murakami, Seren Ozmehmet Tasan, Mitsuo Gen, and Takashi Oyabu. A case study of human resource allocation for effective hotel management. *Industrial Engineering and Management Systems*, 10(1):54–64, 2011.

[81] G Sena Daş and Tolunay Göçken. A fuzzy approach for the reviewer assignment problem. *Computers & industrial engineering*, 72:50–57, 2014.

[82] Naveen Garg, Telikepalli Kavitha, Amit Kumar, Kurt Mehlhorn, and Julián Mestre. Assigning papers to referees. *Algorithmica*, 58(1):119–136, 2010.

[83] Fan Wang, Ning Shi, and Ben Chen. A comprehensive survey of the reviewer assignment problem. *International Journal of Information Technology & Decision Making*, 9(04):645–668, 2010.

[84] Samuel Lukas, Arnold Aribowo, and Milyandreana Muchri. Solving timetable problem by genetic algorithm and heuristic search case study: universitas pelita harapan timetable. *Real-World Applications of Genetic Algorithms*, 378:303–316, 2012.

[85] Thatchai Thepphakorn, Pupong Pongcharoen, and Chris Hicks. An ant colony based timetabling tool. *International Journal of Production Economics*, 149:131–144, 2014.

[86] OA Odeniyi, EO Omidiora, SO Olabiyisi, and JO Aluko. Development of a modified simulated annealing to school timetabling problem. *International Journal of Applied Information Systems*, 8(2):16–24, 2015.

[87] M Güray Güler, M Emre Keskin, Alper Döyen, and Hasan Akyer. On teaching assistant-task assignment problem: A case study. *Computers & Industrial Engineering*, 79:18–26, 2015.

[88] Leh. Luoh and Ruey-Shyang Chang. Solving military's resource allocation problem by fuzzy approach. In *2008 IEEE International Conference on Systems, Man and Cybernetics*, pages 1765–1769, 2008.

[89] Junayed Pasha, Zeinab Elmi, Sumit Purkayastha, Amir M Fathollahi-Fard, Ying-En Ge, Yui-Yip Lau, and Maxim A Dulebenets. The drone scheduling problem: A systematic state-of-the-art review. *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[90] Ma Xian-Ying. Application of assignment model in pe human resources allocation. *Energy Procedia*, 16:1720–1723, 2012.

[91] Antonino Scarelli and Subhash C Narula. A multicriteria assignment problem. *Journal of Multi-Criteria Decision Analysis*, 11(2):65–74, 2002.

[92] Nick Russell, Wil MP van der Aalst, Arthur HM Ter Hofstede, and David Edmond. Workflow resource patterns: Identification, representation and tool support. In *Advanced Information Systems Engineering*, pages 216–232. Springer, 2005.

[93] N. Russell, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. Workflow Resource Patterns. Technical report, BETA Working Paper Series, WP 127, Eindhoven University of Technology, Eindhoven, 2004.

[94] Luise Pufahl, Sven Ihde, Fabian Stiehle, Mathias Weske, and Ingo Weber. Automatic resource allocation in business processes: A systematic literature survey. *CoRR*, abs/2107.07264, 2021.

[95] Michele Lombardi and Michela Milano. Optimal methods for resource allocation and scheduling: a cross-disciplinary survey. *Constraints*, 17:51–85, 2012.

[96] Guidong Zhu, Jonathan F Bard, and Gang Yu. A branch-and-cut procedure for the multimode resource-constrained project-scheduling problem. *INFORMS Journal on Computing*, 18(3):377–390, 2006.

[97] Andreas Drexl, Ruediger Nissen, James H Patterson, and Frank Salewski. Progen/$\pi$x–an instance generator for resource-constrained project scheduling problems with partially renewable resources and further extensions. *European Journal of Operational Research*, 125(1):59–72, 2000.

[98] Rainer Kolisch and Arno Sprecher. Psplib-a project scheduling problem library: Or softwareorsep operations research software exchange program. *European journal of operational research*, 96(1):205–216, 1997.

[99] Raffaele Conforti, Giancarlo Fortino, Marcello La Rosa, and Arthur HM Ter Hofstede. History-aware, real-time risk detection in business processes. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 100–118. Springer, 2011.

[100] Suriadi Suriadi, Burkhard Weiß, Axel Winkelmann, Arthur HM ter Hofstede, Michael Adams, Raffaele Conforti, Colin Fidge, Marcello La Rosa, Chun Ouyang, Anastasiia Pika, et al. Current research in risk-aware business process management—overview, comparison, and gap analysis. *Communications of the Association for Information Systems*, 34(1):52, 2014.

[101] Richard M Zahoransky, Christian Brenig, and Thomas Koslowski. Towards a process-centered resilience framework. In *2015 10th International Conference on Availability, Reliability and Security*, pages 266–273. IEEE, 2015.

[102] Jan vom Brocke and Michael Rosemann. *Handbook on Business Process Management 1: Introduction, Methods, and Information Systems*. Springer, 2015.

[103] Linh Thao Ly, Fabrizio Maria Maggi, Marco Montali, Stefanie Rinderle-Ma, and Wil M.P. van der Aalst. Compliance monitoring in business processes: Functionalities, application, and tool-support. *Information Systems*, 54:209–234, 2015.

[104] Michele Lombardi and Michela Milano. Optimal methods for resource allocation and scheduling: a cross-disciplinary survey. *Constraints*, 17:51–85, 2012.

[105] Michael Rosemann and Jan vom Brocke. The six core elements of business process management. In *Handbook on Business Process Management 1*, pages 105–122. Springer, 2015.

[106] Jan vom Brocke and Michael Rosemann. *Handbook on Business Process Management 2: Strategic Alignment, Governance, People and Culture*. Springer Publishing Company, Incorporated, 2015.

[107] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. *Answer Set*

*Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

[108] Roman Barták. Rina dechter , constraint processing, morgan kaufmann publisher (2003) ISBN 1-55860-890-7, francesca rossi, peter van beek and toby walsh, editors, handbook of constraint programming, elsevier (2006) ISBN 978-0-444-52726-4. *Comput. Sci. Rev.*, 2(2):123–130, 2008.

[109] Simon Tjoa, Stefan Jakoubi, Sigrun Goluch, and Gerhard Kitzler. Planning dynamic activity and resource allocations using a risk-aware business process management approach. In *2010 International Conference on Availability, Reliability and Security*, pages 268–274. IEEE, 2010.

[110] Stefan Jakoubi, Thomas Neubauer, and Simon Tjoa. A roadmap to risk-aware business process management. In *Services Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific*, pages 23–27. IEEE, 2009.

[111] Alan Hevner and Samir Chatterjee. *Design science research in information systems*. Springer, 2010.

[112] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS Q.*, 28(1):75–105, March 2004.

[113] Ken Peffers, Tuure Tuunanen, Marcus A Rothenberger, and Samir Chatterjee. A design science research methodology for information systems research. *Journal of management information systems*, 24(3):45–77, 2007.

[114] Giray Havur, Cristina Cabanillas, Jan Mendling, and Axel Polleres. Automated resource allocation in business processes with answer set programming. In *Business Process Management Workshops: BPM 2015, 13th International Workshops, Innsbruck, Austria, August 31 – September 3, 2015, Revised Papers*, pages 191–203, Cham, 2016. Springer International Publishing.

[115] Giray Havur, Cristina Cabanillas, Jan Mendling, and Axel Polleres. Resource allocation with dependencies in business process management systems. In Marcello La Rosa, Peter Loos, and Oscar Pastor, editors, *Business Process Management Forum*, pages 3–19, Cham, 2016. Springer International Publishing.

[116] Dan Brickley and R.V. Guha. RDF Schema 1.1. W3C Recommendation, February 2014. `http://www.w3.org/TR/rdf-schema/`.

[117] Giray Havur, Cristina Cabanillas, and Axel Polleres. Benchmarking answer set programming systems for resource allocation in business processes. *Expert Systems with Applications*, 205:117599, 2022.

[118] Giray Havur, Cristina Cabanillas, and Axel Polleres. BRANCH: An ASP systems benchmark for resource allocation in business processes. In *Proceedings of the Best Dissertation Award, Doctoral Consortium, and Demonstration & Resources Track at BPM 2021*, volume 2973 of *CEUR Workshop Proceedings*, pages 176–180. CEUR-WS.org, 2021.

[119] Giray Havur. A comparison of ASP and CP solutions for resource allocation in business processes. Technical report, Working Papers on Information Systems, 2022.

[120] Saimir Bala, Cristina Cabanillas, Alois Haselböck, Giray Havur, Jan Mendling, Axel Polleres, Simon Sperl, and Simon Steyskal. A framework for safety-critical process management in engineering projects. In *Data-Driven Process Discovery and Analysis - 5th IFIP WG 2.6 International Symposium, SIMPDA 2015, Vienna, Austria, December 9-11, 2015, Revised Selected Papers*, volume 244 of *Lecture Notes in Business Information Processing*, pages

1–27. Springer, 2015.

[121] Simon Sperl, Giray Havur, Simon Steyskal, Cristina Cabanillas, Axel Polleres, and Alois Haselböck. Resource utilization prediction in decision-intensive business processes. In *Proceedings of the 7th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2017), Neuchâtel, Switzerland, December 6-8, 2017*, volume 2016 of *CEUR Workshop Proceedings*, pages 128–141. CEUR-WS.org, 2017.

[122] Giray Havur and Cristina Cabanillas. History-aware dynamic process fragmentation for risk-aware resource allocation. In *On the Move to Meaningful Internet Systems: OTM 2019 Conferences - Confederated International Conferences: CoopIS, ODBASE, C&TC 2019, Rhodes, Greece, October 21-25, 2019, Proceedings*, volume 11877 of *Lecture Notes in Computer Science*, pages 533–551. Springer, 2019.

[123] Giray Havur, Alois Haselböck, and Cristina Cabanillas. Automated multi-perspective process generation in the manufacturing domain. In *Business Process Management Workshops - BPM 2019 International Workshops, Vienna, Austria, September 1-6, 2019, Revised Selected Papers*, volume 362 of *Lecture Notes in Business Information Processing*, pages 81–92. Springer, 2019.

[124] Saimir Bala, Giray Havur, Simon Sperl, Simon Steyskal, Alois Haselböck, Jan Mendling, and Axel Polleres. SHAPEworks: A BPMS extension for complex process management. In *Proceedings of the BPM Demo Track 2016 Co-located with the 14th International Conference on Business Process Management (BPM 2016), Rio de Janeiro, Brazil, September 21, 2016*, volume 1789 of *CEUR Workshop Proceedings*, pages 50–55. CEUR-WS.org, 2016.

[125] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Max Ostrowski, Torsten Schaub, and Sven Thiele. Engineering an incremental ASP solver. In Maria Garcia de la Banda and Enrico Pontelli, editors, *Logic Programming, 24th International Conference, ICLP 2008, Udine, Italy, December 9-13 2008, Proceedings*, volume 5366 of *Lecture Notes in Computer Science*, pages 190–205. Springer, 2008.

[126] Francesco Calimeri, Wolfgang Faber, Martin Gebser, Giovambattista Ianni, Roland Kaminski, Thomas Krennwallner, Nicola Leone, Francesco Ricca, and Torsten Schaub. ASP-Core-2: Input language format. Technical report, ASP Standardization Working Group, 2013.

[127] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. *Answer Set Solving in Practice*. Morgan & Claypool Publishers, 2012.

[128] Francesca Rossi, Peter Van Beek, and Toby Walsh. *Handbook of constraint programming*. Elsevier, 2006.

[129] Nicholas Nethercote, Peter J Stuckey, Ralph Becket, Sebastian Brand, Gregory J Duck, and Guido Tack. Minizinc: Towards a standard cp modelling language. In *International Conference on Principles and Practice of Constraint Programming*, pages 529–543. Springer, 2007.

Published in the Proceedings of the 13th Int. Conference on Business Process Management Workshops (BPM 2015 International Workshops), pp. 191-203, Jul 2016, Springer LNBIP vol. 256

# Automated Resource Allocation in Business Processes with Answer Set Programming*

Giray Havur, Cristina Cabanillas, Jan Mendling, and Axel Polleres

Vienna University of Economics and Business, Austria
{firstname.lastname}@wu.ac.at

**Abstract.** Human resources are of central importance for executing and supervising business processes. An optimal resource allocation can dramatically improve undesirable consequences of resource shortages. However, existing approaches for resource allocation have some limitations, e.g., they do not consider concurrent process instances or loops in business processes, which may greatly alter resource requirements. This paper introduces a novel approach for automatically allocating resources to process activities in a time optimal way that is designed to tackle the aforementioned shortcomings. We achieve this by representing the resource allocation problem in Answer Set Programming (ASP), which allows us to model the problem in an extensible, modular, and thus maintainable way, and which is supported by various efficient solvers.

**Keywords:** Answer Set Programming, business process management, resource allocation, timed Petri net, work scheduling

## 1 Introduction

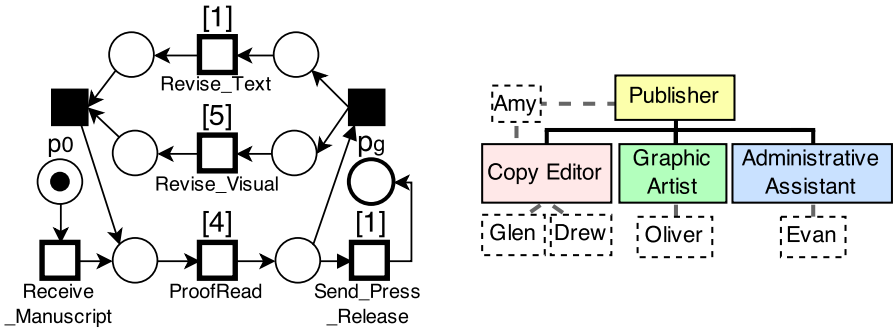
Human resources[1] are crucial in business process management (BPM) as they are responsible for process execution or supervision. A lack of resources or a suboptimal work schedule may produce delayed work, potentially leading to a reduced quality and higher costs.

In this paper, we address the problem of allocating the resources available in a company to the activities in the running process instances in a time optimal way, i.e., such that process instances are completed in the minimum amount of time. Our approach lifts limitations of prior research pursuing similar goals, which assumes simplified non-cyclic processes and does not necessarily search for an optimal resource allocation [16, 14]. To this end, we rely on Answer Set Programming (ASP) [6], a declarative knowledge representation and reasoning formalism that is supported by a wide range of efficient solvers. ASP has been successfully used to address planning and configuration problems in other domains [5].

Our solution is divided into three layers: The core layer represents process models in ASP. The second layer adds all the information related to time, such as the estimated activity durations. Finally, resource-related information including, among others, the

* Funded by the Austrian Research Promotion Agency (FFG), grant 845638 (SHAPE).

[1] From now on *resources* for the sake of brevity.




(a) Process to publish a book

(b) Organisational model

Fig. 1: Running example

characteristics of the resources available according to an organisational model as well as the conditions that must be fulfilled to assign resources to activities (e.g., to have a specific organisational role), is encoded on top of these two layers. An ASP solver can use all this data to compute possible optimal solutions for the resource allocation problem. We have evaluated our approach with a proof-of-concept implementation and we have measured its performance with non-trivial scenarios that contain loops and concurrent process instances.[2]

Our modular encodings in ASP provide flexibility and extensibility so that, e.g., additional instances of pre-defined processes can be added. In addition, the declarative nature of the encodings of constraints enables an executable specification of the problem.

The paper is structured as follows: Section 2 presents a scenario that motivates this work as well as related work. Section 3 defines technical background required to understand our approach. Section 4 describes our modular approach for resource allocation in business processes with ASP. Section 5 presents the evaluations performed and Section 6 concludes and outlines future work.

## 2 Background

In the following, we describe an example scenario that motivates this work and shows the problems to be addressed, and then we outline related work.

### 2.1 Running Example

In this paper we rely on (timed) Petri nets [12] for business process modelling, commonly used for this purpose due to their well defined semantics and their analysis capabilities. Nonetheless, any process modelling notation can be used with our approach as long as it can be mapped to Petri nets, for which several transformations have already

[2] Our encoding and the problem instances are provided at `http://goo.gl/lzf1St`

been defined [11]. Fig. 1a depicts a model representing the process of publishing a book from the point of view of a publishing entity. In particular, when the publishing entity receives a new textbook manuscript from an author, it must be proofread. If changes are required, the modifications suggested must be applied on text and figures, which can be done in parallel. This review-and-improvement procedure is repeated until there are no more changes to apply, and the improved manuscript is then sent back to the author for double-checking. In Fig. 1a, the numbers above the activities indicate their (default maximum) duration in generic time units (TU)[3].

The organisational model depicted in Fig. 1b shows the hierarchy of roles of a publishing entity. Specifically, it has four roles and five resources assigned to them. The following relation specifies how long it takes to each role and resource to complete the process activities: $(Role \cup Resource) \times Activity \times TU \supset$\{(Copy Editor, Proofread, 2), (Glen, Proofread, 5), (Drew, Proofread, 2), (Drew, Revise Text, 2)\}. For resource allocation purposes, the duration associated with a specific resource is used in first place followed by the duration associated with roles and finally, the duration of activities (cf. Fig. 1a). Resources are assigned to activities according to their roles. In particular, the relation activity-role in this case is as follows: $Role \times Activity \supset$\{(Publisher, Receive Manuscript), (Copy Editor, Proofread), (Copy Editor, Revise Text), (Graphic Artist, Revise Visual), (Admin. Asst., Send Press Release)\}.

For the purpose of planning the allocation of resources to process activities in an optimal way, the following aspects must be taken into consideration: (i) several process instances can be running at the same time; (ii) the review-and-improvement procedure is a loop and hence, it may be repeated several times in a single process instance. Since one cannot know beforehand the number of repetitions that will be required for each process instance, assumptions must be made about it. Optimality is reached when the activities in all instances of a business process are assigned resources so that the overall execution of all instances takes as little time as possible.

### 2.2 Related Work

The existing work on resource allocation in business processes has mostly relied on Petri nets. In fact, the goal we pursue is doable at the Petri net level with some shortcomings and limitations. Van der Aalst [16] introduced a Petri net based scheduling approach to show that the Petri net formalism can be used to model activities, resources and temporal constraints with non-cyclic processes. However, modelling this information for multiple process instances leads to very large Petri nets. Moreover, other algorithms for resource allocation proved to perform better than that approach [3]. Rozinat et al. [14] used Coloured Petri nets (CPNs) to overcome the problems encountered in traditional Petri nets. In CPNs, classes and guards can be specified to define any kind of constraints. However, the approach proposed is greedy such that resources are allocated to activities as soon as they are available, overlooking the goal of finding an optimal solution. This may make the allocation problem unsatisfiable.

Several attempts have also been done to implement the problem as a constraint satisfaction problem. For instance, Senkul and Toroslu [15] developed an architecture

[3] Please, note that events are instantaneous, and hence, they take zero time units.

to specify resource allocation constraints and a Constraint Programming (CP) approach to schedule a workflow according to the constraints defined for the tasks. However, they aimed at obtaining a feasible rather than an optimal solution and the approach does not support the schedule of concurrent workflows. Besides, Heinz and Beck [7] demonstrated that models such as Constraint Integer Programming (CIP) outperform the standard CP formulations. In addition, loops are disregarded in these approaches.

Resource allocation in projects has been widely investigated [17, 8]. However, projects differ from business processes in that they are typically defined to be executed only once and decision points are missing. Therefore, the problem is approached in a different way. The agent community has also studied how to distribute a number of resources among multiple agents [4, 19]. Further research in necessary to adapt those results to resource allocation in business processes [18].

## 3 Preliminaries

**Timed Petri Nets** [13] associate durations with transitions: a *timed Petri net* is a 5-tuple $N_T = \langle P, T, F, c, M_0 \rangle$ such that $P$ is a finite set of *places*, $T$ is a finite set of *transitions*, with $P \cap T = \emptyset$, $F \subset (P \times T) \cup (T \times P)$ describes a bipartite graph, $M_0$ is the *initial marking*. and $c : T \to \mathbb{N}$ is a function that assigns firing delays to every transition $t \in T$. Here, a *marking*(state) $M : P \to \mathbb{Z}^+$ assigns to each place a non-negative integer, denoting number of tokens in places. For each $t \in T$ the *input place set* $\bullet t = \{p \in P \mid (p, t) \in F\}$. The output place set $t\bullet$, and analogously input $\bullet p$ (and output $p\bullet$,resp.) transition sets of a place $p \in P$ can be defined analogously. A transition may *fire*, written $\xrightarrow{t}$, when all $p \in \bullet t$ have tokens: all tokens in $\bullet t$ are consumed and tokens produced in each $p \in t\bullet$.

A Fig. 1a shows an example of a timed Petri net: circles represent places, squares represent transitions, and numbers in brackets on transitions denote firing delays. Filled squares denote "silent" transitions that have no firing delays, i.e., $c(t) = 0$. However, note that also normal transitions that correspond to activities can have no delay, e.g., $t_m$ in Fig. 1a.

A marking $M_k$ is *reachable* from $M_{k-1}$ in *one step* if $M_{k-1} \xrightarrow{t_{k-1}} M_k$. A firing sequence of transitions $\overrightarrow{\sigma} = \langle t_1 t_2 ... t_n \rangle$ changes the state of the Petri net at each firing: $M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 \xrightarrow{\cdots} M_n$. In this paper we use *1-safe Petri nets*, i.e., each place contains at most one token in any state. $N_T$ is called *sound* if from every reachable state, a proper final state can be reached in $N$. $N_T$ is called *free-choice* if every for transitions $t_1$ and $t_2$, $\bullet t_1 \cap \bullet t_2 \neq \emptyset$ implies $\bullet t_1 = \bullet t_2$.

**Answer Set Programming (ASP)** [1, 6] is a declarative (logic-programming-style) paradigm for solving combinatorial search problems

An *ASP program* $\Pi$ is a finite set of rules of the form

$$A_0 \text{ :- } A_1, \ldots, A_m, \textit{not } A_{m+1}, \ldots, \textit{not } A_n. \tag{1}$$

where $n \geq m \geq 0$ and each $A_i \in \sigma$ are (function-free first-order) atoms; if $A_0$ is empty in a rule $r$, we call $r$ a constraint, and if $n = m = 0$ we call $r$ a fact. Whenever $A_i$ is a

first-order predicate with variables within a rule of the form (1), this rule is considered as a shortcut for its "grounding" $ground(r)$, i.e., the set of its ground instantiations obtained by replacing the variables with all possible constants occurring in $\Pi$. Likewise, we denote by $ground(\Pi)$ the set of rules obtained from grounding all rules in $\Pi$.

Sets of rules are evaluated in ASP under the so-called stable-model semantics, which allows several models (so called "answer sets"), that is subset-minimal Herbrand models, we again refer to [1] and references therein for details.

ASP Solvers typically first compute (a subset of $ground(\Pi)$, and then use a DPLL-like branch and bound algorithm is used to find answer sets for this ground program. There are various solvers [2, 9] for ASP problem specifications, we use $clasp$ [6] for our experiments herein (cf. Section 5), as one of the most efficient implementations available.

As syntactic extension, in place of atoms, $clasp$ allows set-like *choice expressions* of the form $E = \{A_1, \ldots, A_k\}$ which are true for any subset of $E$; that is, when used in heads of rules, $E$ generates many answer sets, and such rules are often referred to as *choice rules*. For instance, $\Pi_4 = \{lights\_on.\{shop\_open, door\_locked\} :- lights\_on.\}$ has both answer sets of $\Pi_3$ plus the answer set $\{lights\_on\}$. Note that in the presence of choice rules, answer sets are not necessarily subset-minimal, we refer to [6] for details.

Another extension supported in $clasp$ are optimisation statements [6] to indicate preferences between possible answer sets:

$$\#minimize\ \{A_1 : Body_1 = w_1, \ldots, A_m : Body_m = w_m\}$$

associates integer weights (defaulting to 1) with atoms $A_i$ (conditional to $Body_i$ being true), where such a statement expresses that we want to find only answer sets with the smallest aggregated weight sum; again, variables in $A_i : Body_i = w_i$ are replaced at grounding w.r.t. all possible instantiations.

Finally, many problems conventiently modelled in ASP require a boundary parameter $k$ that reflects the size of the solution. However, often in problems like planning or model checking this boundary (e.g., the plan length) is not known upfront, and therefore such problems are addressed by considering one problem instance after another while gradually increasing this parameter $k$. However, re-processing repeatedly the entire problem is a redundant approach, which is why incremental ASP (iASP) [6] natively supports incremental computation of answer sets; the intuition is rooted in treating programs in program slices (extensions). In each incremental step, a successive extension of the program is considered where previous computations are re-used as far as possible.

An iASP program is a triple $(B, A[k], Q[k])$, where $B$ describes *static* knowledge, and $A[k]$ and $Q[k]$ are ASP programs parameterized by the incremental parameter $k \in \mathbb{N}^+$. In the iterative answer set computation of iASP, while the knowledge derived from the rules in $A[k]$ accumulates as $k$ increases, the knowledge obtained from $Q[k]$ is only considered for the latest value of $k$. $A[k]$ and $Q[k]$ are called *cumulative* knowledge and *volatile* knowledge, resp. More formally, an iASP solver computes in each iteration $i$

$$\Pi[i] = B \cup \bigcup_{1 \leq j \leq i} A[k/j] \cup Q[k/i]$$

until an answer set for some (minimum) integer $i \geq 1$ is found. We will demonstrate next, how iASP can be successfully used to model and solve various variants of resource allocation problems in business process management.

## 4 Resource Allocation with iASP

For tackling the problem of resource allocation in business processes, we have developed a modular iASP program consisting of three layers. The bottom layer is the generic iASP encoding $\Pi_N$ for finding a firing sequence between initial and goal markings of a 1-safe Petri net $N$. This provides a marking of $N$ at each value of parameter $k$. On a second layer we extend $\Pi_N$ towards $\Pi_T$ to encode timed Petri Nets, i.e., we support business processes encoded as timed Petri nets whose activities can have a duration. Consequently, this encoding cannot only compute possible markings, but also the overall duration for a firing sequence. In other words, now we also know about the value of the overall time spent time at a firing sequence of length $k$. In the upper layer $\Pi_R$, we include rules and constraints about resources in order to encode an iASP program that allocates activities to available resources for a certain period of time.

Please, note some general assumptions that we make about the structure of a resource allocation problem: (i) no resource may process more than one activity at a time; (ii) each resource is continuously available for processing; (iii) no pre-emption, i.e., each activity, once started, must be completed without interruptions; and (iv) the processing times are independent of the schedule, and they are known in advance. These assumptions are common in related approaches [16].

### 4.1 $\Pi_N$ : A Generic Formulation of 1-safe Petri Nets

Based on the notions introduced in Section 3, we formalise the firing dynamics of 1-safe Petri net $N = \langle P, T, F, M_0 \rangle$ in an iASP program $(B_N, A_N[k], Q_N[k])$. Given a goal state $M_k$, which for the sake of simplicity we assume to be defined in terms of a single goal place $p_g$, the aim is to find a shortest possible firing sequence $\overrightarrow{\sigma} = \langle t_1 t_2 ... t_k \rangle$ that does not violate the constraints, from $M_0$ to $M_k$.

**$B_N$:** $N = \langle P, T, F, M_0 \rangle$ is represented using predicates $\texttt{inPlace}_N(\texttt{p}, \texttt{t})$ and $\texttt{outPlace}_N(\texttt{p}, \texttt{t})$ that encode $F$. We encode different instances $i$ of $N$ by the predicate $\texttt{instance}_N$, which allows us to run the allocation problem against different instances of the same process; initial markings of instance $M_{0_i}$ are defined via predicate $\texttt{tokenAt}_N(\texttt{P}_0, \texttt{k}_0, \texttt{i})$ where for each $p \in P_0$, $M_0(p) = 1$.[4]

**$A_N[k]$:** is shown in Fig. 2. Rule (2) guesses all subsets of possible firing actions for each instance of $N$. Constraint (3) ensures that any transition $t \in \texttt{T}$ is fired only if all input places in $\bullet t$ have tokens. Rule (4) models the effect of the action `fire` on output places by assigning a token to each output place in the step following the firing. Constraint (5) prohibits concurrent firings of transitions $t \in p\bullet$. Rules (6) and (7) preserve tokens at place $p$ in successive steps if none of the transitions $t \in p\bullet$ fires.

[4] Since in the following we only consider instances of the same Petri Net, we will drop the subscript $N$ in the predicates.

$A_N[k]$ :

$\{\texttt{fire(T, k, I)} : \texttt{inPlace(P, T)}, \texttt{instance(I)}\}.$ (2)

$\texttt{:-fire(T, k, I)}, \texttt{instance(I)}, \texttt{inPlace(P, T)}, \texttt{not tokenAt(P, k, I)}.$ (3)

$\texttt{tokenAt(P, k, I) :-fire(T, k} - 1, \texttt{I)}, \texttt{outPlace(P, T)}, \texttt{instance(I)}.$ (4)

$\texttt{:-inPlace(P, T1)}, \texttt{inPlace(P, T2)}, \texttt{T1!} = \texttt{T2}, \texttt{fire(T1, k, I)}, \texttt{fire(T2, k, I)},$ (5)
$\texttt{instance(I)}.$

$\texttt{consumeToken(P, k, I) :-inPlace(P, T)}, \texttt{fire(T, k, I)}, \texttt{instance(I)}.$ (6)

$\texttt{tokenAt(P, k, I) :-tokenAt(P, k} - 1, \texttt{I)}, \texttt{not consumeToken(P, k} - 1, \texttt{I)}.$ (7)

$Q_N[k]$ :

$\texttt{:-not tokenAt(p}_\texttt{g}\texttt{, k, I)}, \texttt{instance(I)}.$ (8)

Fig. 2: 1-safe Petri net formulation in iASP

$\boldsymbol{Q_N[k]}$**:** Finally, constraint (8) in Fig. 2 enforces a token to reach the goal place $p_g$ (for all instances $i \in I$). The computation ends as soon as this constraint is not violated in an iteration of the iASP program, i.e., it computes the minimally necessary number of iterations $k$ to reach the goal state.

### 4.2 $\Pi_T$ : Activity Scheduling using Timed Petri Net

In order to model activity durations, we extend the above iASP encoding towards Timed Petri nets: that is, $\Pi_N$ is enhanced with the notion of time in $\Pi_T$. By doing so, $\Pi_N \cup \Pi_T$ becomes capable of scheduling activities in instances of a timed Petri net $N_T$.

$\boldsymbol{B_T}$**:** We expand the input of $\Pi_N$ with facts related to time and with the rules that are independent from the parameter $k$. For each fact $\texttt{tokenAt(p}_0\texttt{, k}_0\texttt{, i)}$ previously defined we add in $B_T$ a fact $\texttt{timeAt(p}_0\texttt{, c}_0\texttt{, k}_0\texttt{, i)}$ where $c_0$ is the initial time at $p_0$. In order to distinguish activity transitions and ("silent") non-activity transitions[5], we add facts `activity(t)` for all activities. Durations of activities are specified with facts `timeActivity(t,c)` where $t$ is an activity and $c \in \mathbb{Z}^+$. The remainder of $B_T$ is given by rules (9,10) in Fig. 3: rule (9) defines firing delays of each transition in $N$ and rule (10) assigns duration zero to activity transitions per default, where the delay is not otherwise specified.

$\boldsymbol{A_T[k]}$**:** Rule (13) defines the effect of action `fire` on `timeAt` for all output places $t\bullet$ where $t$ is a *non-activity* transition. In this case, the maximum time among the input places, which is computed by rules (11,12), is propagated over all output places. As opposed to (13), rule (14) defines the effect of action `fire` on `timeAt` for *activity* transitions. Time value derived in rule (14) for the next step is the sum of the maximum time value at the input places and the value of the activity duration. Rule (15) conserves the time value of a place in the succeeding step $k$ in case the transition does not fire at step $k - 1$.

[5] Recall: in Petri nets representing business processes, activity transitions are empty squares while silent transitions are represented in filled squares (cf. Fig. 1a).

$B_T$ :

$\texttt{firingDelay(T,C)} \texttt{:-timeActivity(T,C)}.$ (9)

$\texttt{firingDelay(T,0)} \texttt{:-not timeActivity(T,\_), activity(T)}.$ (10)

$A_T[k]$ :

$\texttt{greTimeInPlace(P1,T,k,I)} \texttt{:-inPlace(P1,T), inPlace(P2,T), fire(T,k,I),}$ (11)
$\texttt{timeAt(P1,C1,k,I), timeAt(P2,C2,k,I), P1!} = \texttt{P2,}$
$\texttt{C1} < \texttt{C2, instance(I)}.$

$\texttt{maxTimeInPlace(P,T,k,I)} \texttt{:-inPlace(P,T), not greTimePlace(P,T,k,I),}$ (12)
$\texttt{fire(T,k,I), instance(I)}.$

$\texttt{timeAt(P2,C,k,I)} \texttt{:-not activity(T), fire(T,k} - 1\texttt{,I), outPlace(P2,T),}$ (13)
$\texttt{maxTimeInPlace(P,T,k} - 1\texttt{,I), timeAt(P,C,k} - 1\texttt{,I),}$
$\texttt{instance(I)}.$

$\texttt{timeAt(P2,C1,k,I)} \texttt{:-activity(T), fire(T,k} - 1\texttt{,I), outPlace(P2,T),}$ (14)
$\texttt{maxTimeInPlace(P,T,k} - 1\texttt{,I), timeAt(P,C,k} - 1\texttt{,I),}$
$\texttt{firingDelay(T,D), C1} = \texttt{C} + \texttt{D, instance(I)}.$

$\texttt{timeAt(P,C,k,I)} \texttt{ :-not consumeToken(P,k} - 1\texttt{,I), inPlace(P,T),}$ (15)
$\texttt{timeAt(P,C,k} - 1\texttt{,I), instance(I)}.$

$Q_T[k]'$ :

$\#\texttt{minimize}\{\texttt{timeAt}(\texttt{p}_\texttt{g}\texttt{,C,k,I}) : \texttt{instance(I)} = \texttt{C}\}$ (16)

Fig. 3: Scheduling extension

$\boldsymbol{Q_T[k]}$**:** On top of $Q_N[k]$, an optimization statement (16) is added for computing answer sets with the minimum time cost.

### 4.3 $\boldsymbol{\Pi_R}$ : Resource Allocation

In the last layer of our iASP program, $\Pi_R$, we additionally formalise resources and related concepts. $\Pi_N \cup \Pi_T \cup \Pi_R$ allow allocating resources to activities for a time optimal execution of all defined instances of $N_T$.

$\boldsymbol{B_R}$**:** The facts related to resources and organisational models are defined in the input of $\Pi_T$. An example organisational model is shown in Fig. 1b. Facts `hasRole(r,l)` relates a resource $r$ to a role $l$. Activities are related to a role via facts of the form `canExecute(l,t)`, which means that a role $l$ is allowed to performing an activity $t$. An optional estimated duration for a resource to execute an activity can be defined by `timeActivityResource(t,r,c)`. Similarly an optional estimated duration for a role per activity can be defined by `timeActivityRole(t,l,c)`. Both can override the default `timeActivity(t,c)`. In particular, the order ($>$) preferred in resource-time allocation is `timeActivityResource` $>$ `timeActivityRole` $>$ `timeActivity`. This is especially useful when a resource or a role is known to execute a particular activity in a particular amount of time, which can be different from the default duration of the activity. In our program (cf. Fig. 4) this preference computation is encoded in rules (17-21).

$B_R$ :

$\texttt{existsTimeActivityResource}(\texttt{T},\texttt{R}) \texttt{:-} \texttt{timeActivityResource}(\texttt{T},\texttt{R},\texttt{C}).$ (17)

$\texttt{existsTimeActivityRole}(\texttt{T},\texttt{L}) \texttt{:-} \texttt{timeActivityRole}(\texttt{T},\texttt{L},\texttt{C}), \texttt{hasRole}(\texttt{R},\texttt{L}).$ (18)

$\texttt{takesTime}(\texttt{T},\texttt{R},\texttt{C}) \texttt{:-} \texttt{timeActivityResource}(\texttt{T},\texttt{R},\texttt{C}).$ (19)

$\texttt{takesTime}(\texttt{T},\texttt{R},\texttt{C}) \texttt{:-} \texttt{timeActivityRole}(\texttt{T},\texttt{L},\texttt{C}), \texttt{hasRole}(\texttt{R},\texttt{L}), \texttt{canExecute}(\texttt{L},\texttt{T}),$ (20)
$\texttt{not existsTimeResource}(\texttt{T},\texttt{R}).$

$\texttt{takesTime}(\texttt{T},\texttt{R},\texttt{C}) \texttt{:-} \texttt{firingDelay}(\texttt{T},\texttt{C}), \texttt{hasRole}(\texttt{R},\texttt{L}), \texttt{canExecute}(\texttt{L},\texttt{T}),$ (21)
$\texttt{not existsTimeActivityResource}(\texttt{T},\texttt{R}),$
$\texttt{not existsTimeActivityRole}(\texttt{T},\texttt{L}).$

$A_R[k]$ :

$\{\texttt{assign}(\texttt{R},\texttt{T},\texttt{C},\texttt{C2},\texttt{k},\texttt{I}) : \texttt{takesTime}(\texttt{T},\texttt{R},\texttt{C}), \texttt{C2} = \texttt{C} + \texttt{D}\} \texttt{:-} \texttt{inPlace}(\texttt{P1},\texttt{T}),$ (22)
$\texttt{timeAt}(\texttt{P1},\texttt{C},\texttt{k},\texttt{I}), \texttt{activity}(\texttt{T}), \texttt{instance}(\texttt{I}).$

$\texttt{timeAt}(\texttt{P2},\texttt{C2},\texttt{k},\texttt{I}) \texttt{:-} \texttt{activity}(\texttt{T}), \texttt{assign}(\texttt{R},\texttt{T},\texttt{C1},\texttt{C2},\texttt{k}-1,\texttt{I}),$ (14)*
$\texttt{fire}(\texttt{T},\texttt{k}-1,\texttt{I}), \texttt{outPlace}(\texttt{P2},\texttt{T}), \texttt{instance}(\texttt{I}).$

$\texttt{assigned}(\texttt{T},\texttt{k},\texttt{I}) \texttt{:-} \texttt{assign}(\texttt{R},\texttt{T},\texttt{C1},\texttt{C2},\texttt{k},\texttt{I}).$ (23)

$\texttt{:-not assigned}(\texttt{T},\texttt{k},\texttt{I}), \texttt{fire}(\texttt{T},\texttt{k},\texttt{I}), \texttt{activity}(\texttt{T}), \texttt{instance}(\texttt{I}).$ (24)

$\texttt{:-assign}(\texttt{R},\texttt{T},\texttt{C1},\texttt{C2},\texttt{K},\texttt{I}), \texttt{assign}(\texttt{R1},\texttt{T},\texttt{C3},\texttt{C4},\texttt{K},\texttt{I}), \texttt{R!} = \texttt{R1}.$ (25)

$\texttt{:-assign}(\texttt{R},\texttt{T1},\texttt{C1},\texttt{C2},\texttt{K1},\texttt{I1}), \texttt{assign}(\texttt{R},\texttt{T2},\texttt{C1},\texttt{C2},\texttt{K2},\texttt{I2}), \texttt{C1!} = \texttt{C2}, \texttt{T1!} = \texttt{T2}.$ (26)

$\texttt{:-assign}(\texttt{R},\texttt{T},\texttt{C1},\texttt{C2},\texttt{K1},\texttt{I1}), \texttt{assign}(\texttt{R},\texttt{T},\texttt{C1},\texttt{C2},\texttt{K2},\texttt{I2}), \texttt{C1!} = \texttt{C2}, \texttt{I1!} = \texttt{I2}.$ (27)

$\texttt{:-assign}(\texttt{R},\texttt{T1},\texttt{C1},\texttt{C2},\texttt{K1},\texttt{I1}), \texttt{assign}(\texttt{R},\texttt{T2},\texttt{C1},\texttt{C2},\texttt{K2},\texttt{I2}),$ (28)
$\texttt{C1!} = \texttt{C2}, \texttt{I1!} = \texttt{I2}, \texttt{T1!} = \texttt{T2}.$

$\texttt{:-assign}(\texttt{R},\texttt{T},\texttt{B1},\texttt{B2},\texttt{K1},\texttt{I}), \texttt{assign}(\texttt{R},\texttt{T2},\texttt{A1},\texttt{A2},\texttt{K2},\texttt{I2}), \texttt{A1} > \texttt{B1}, \texttt{A1} < \texttt{B2}.$ (29)

$\texttt{:-assign}(\texttt{R},\texttt{T},\texttt{B1},\texttt{B2},\texttt{K1},\texttt{I}), \texttt{assign}(\texttt{R},\texttt{T2},\texttt{A1},\texttt{A2},\texttt{K2},\texttt{I2}), \texttt{A2} < \texttt{B2}, \texttt{A2} > \texttt{B1}.$ (30)

Fig. 4: Allocation extension

Rules (17,18) are projections of optionally defined activity execution durations. Rules (19-21) derive correct execution duration for resource-activity pairs considering both mandatory and optional durations.

$\boldsymbol{A_R[k]}$**:** In the iterative part, rule (22) allocates a resource $r$ to an activity $t$ from time $c$ to time $c2$. Note that, for handling optional execution durations, rule (14) from Fig. 3 is replaced by rule (14)*. Rule (23) along with constraint (24) prohibits any firing of an activity transition that is not allocated to a resource. Constraint (25) ensures that an activity cannot be assigned to more than one resource. Constraints (26-28) guarantee that only one resource is assigned to one activity at a time. Constraints (29,30) prevents a busy resource to be re-assigned.

**Time Relaxation** In case a resource is busy at the time when s/he is required for another activity, our program would be unsatisfiable as it is. We add rules (31) and (32) (cf. Fig. 5) into $A_T[k]$ for allowing the demanding activity to wait until the required resource is available again.

$A_T[k]'$ :

$$\texttt{relaxationAt}(\texttt{P}, \texttt{C}+1, \texttt{k}, \texttt{I}) \texttt{:-} \texttt{timeAt}(\texttt{P}, \texttt{C}, \texttt{k}-1, \texttt{I}), \texttt{inPlace}(\texttt{P}, \texttt{T}), \texttt{activity}(\texttt{T}), \quad (31)$$
$$\texttt{not consumeToken}(\texttt{P}, \texttt{k}-1, \texttt{I}), \texttt{instance}(\texttt{I}).$$

$$\texttt{timeAt}(\texttt{P}, \texttt{C}, \texttt{k}, \texttt{I}) \texttt{:-} \texttt{relaxationAt}(\texttt{P}, \texttt{C}, \texttt{k}, \texttt{I}). \quad (32)$$

Fig. 5: Time relaxation for optimality

# 5 Evaluation

We demonstrate the applicability and effectiveness of the proposed computational method for resource allocation in business processes by using it with a specific process. In order to measure performance and scalability, we conduct a batch experiment using generated examples of timed Petri nets of different sizes.

## 5.1 Example Scenario

We apply our method to a business process model that specifies the process of publishing a book as described in Section 2.1. The input of the program encoded in ASP following the explanations in Section 4 is: (i) three different instances $i1$, $i2$, $i3$ of the timed Petri net depicted in Fig. 1a, whose starting times are defined as $t_{0i1} = 0$, $t_{0i2} = 6$ and $t_{0i3} = 11$, respectively; (ii) the organisational model and optional activity times for resources and roles as shown in Fig. 1b, (iii) role-activity relation defined in Section 2. We also add additional constraints for enforcing the firing sequence to go through the loop present in the process two, three and one times for $i1$, $i2$ and $i3$, respectively.

The computed optimal resource allocation is visualised in Fig. 6. The allocation periods are depicted as coloured rectangles with a tag on it. Each tag has three parts: an initial with the initials of a resource, a short version of the allocated activity name and a subscript representing the instance ID. For example, $D : PR_1$ means that *Drew* is allocated to activity *Proofreading*. The colours of these rectangles correspond to the colours used for the roles depicted in Fig. 1b. Note that *Amy* has more than one roles in the organisation.

The longest process instance $i2$ finishes in 36 time units. Several solutions were found for that global minimum time. In Fig. 6, instances 1 and 2 finish without interruptions. However, instance 3 waits 7 time units for the availability of *Glen* to start performing activity *Proofread*, since he is busy performing that activity for process instance 2 until time unit 23. All-in-all, this computation optimises the use of resource *Oliver*, who is the only *Graphic Artist* and is required in all the process instances. Please, note that, e.g., in instance 2 *Drew* is selected to perform activity *Proofread* because it takes him only 2 time units (cf. Fig. 1b), half of the default duration associated with the activity (cf. Fig. 1a). This responds to the preference order described in Section 4.3.

## 5.2 Performance

For our experimental evaluation, we generated a set of sound choice-free timed Petri nets (cf. Section 3). We varied the number of existing loops in these Petri nets and the

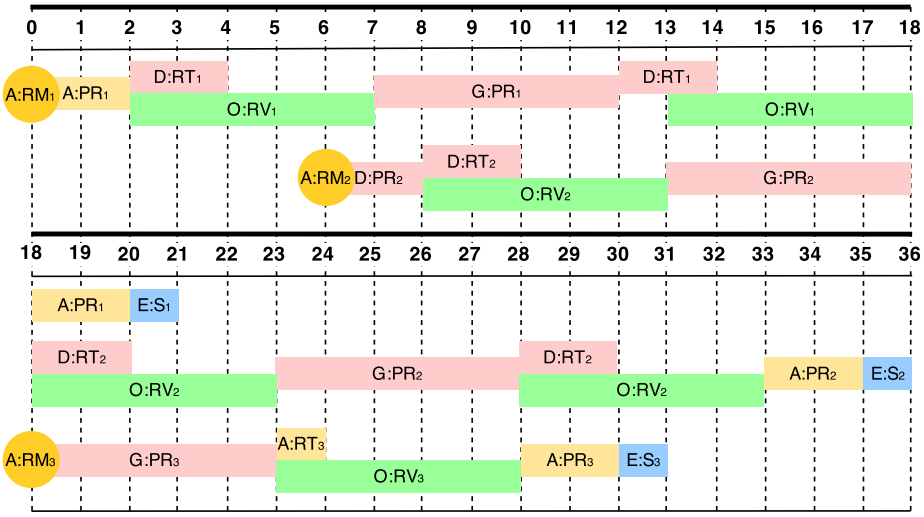



Fig. 6: Instance I - 2 loop repetitions / instance II - 3 loop repetitions / instance III - 1 loop repetition

number of parallel process instances. We use the same organisational model for all of the generated Petri nets, specifically the one depicted in Fig. 1b. We performed these experiments on a Linux server (4 CPU cores/2.4GHz/32GB RAM). *clasp* was used as ASP solver with the multi-threading mode enabled.

The results are shown in Table 1 in two parts. In the programs on the left hand side (1-9), no transitions in the loops are enforced to be fired. In the programs on the right hand side (10-18), each loop in the Petri net is constrained to be followed at least one time. The columns of the table are as follows: $id$ is the identifier of a generated program, $|I|$ is the number of parallel instances, $|L|$ is the number of loops, $|f(T)|$ is the number of fired transitions from initial to goal state, $k$ is the final value of that parameter, $s$ is the time in seconds to find the answer set of the program, and $m$ is the maximum memory usage in megabytes.

For instance, it takes the solver 1.13 seconds to find an answer set for a Petri net with one loop that is not enforced at run time, and 15.02 seconds for a similar Petri net in which the loop is executed. This is satisfactory for many planning scenarios with large processes, as they can be scheduled in a few seconds/minutes and executed for a long period of time.

## 6 Conclusions and Future Work

We have introduced an approach for automated resource allocation in business processes that relies on ASP to find an optimal solution. The result is a work distribution (i.e., an activity allocation) that ensures that all the process activities can finish in the minimum amount of time given a set of resources. Unlike similar approaches, it is capable of dealing with cyclic processes and concurrent process instances as our encoding

| $id$ | $\|I\|$ | $\|L\|$ | $\|f(T)\|$ | $k$ | $s$ | $m$ |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 10 | 8 | 1.13 | 10.2 |
| 2 | 2 | 1 | 20 | 21 | 7.38 | 72.2 |
| 3 | 3 | 1 | 38 | 9 | 176.45 | 432.1 |
| 4 | 1 | 2 | 10 | 3 | 0.57 | 0 |
| 5 | 2 | 2 | 20 | 21 | 83.03 | 459.4 |
| 6 | 3 | 2 | 42 | 31 | 199.46 | 756.8 |
| 7 | 1 | 3 | 10 | 11 | 1.27 | 17.9 |
| 8 | 2 | 3 | 20 | 16 | 28.57 | 229 |
| 9 | 3 | 3 | 38 | 21 | 85.73 | 475.1 |

| $id$ | $\|I\|$ | $\|L\|$ | $\|f(T)\|$ | $k$ | $s$ | $m$ |
|---|---|---|---|---|---|---|
| 10 | 1 | 1 | 24 | 4 | 15.02 | 101.6 |
| 11 | 2 | 1 | 48 | 25 | 90.87 | 419 |
| 12 | 3 | 1 | 72 | 33 | 193.72 | 372.9 |
| 13 | 1 | 2 | 28 | 29 | 33.96 | 186.2 |
| 14 | 2 | 2 | 60 | 7 | 1314.73 | 2877.2 |
| 15 | 3 | 2 | n/a | n/a | 10800 | 5744.1 |
| 16 | 1 | 3 | 24 | 25 | 17.5 | 83.9 |
| 17 | 2 | 3 | 48 | 28 | 161.15 | 496.5 |
| 18 | 3 | 3 | 96 | 4 | 2366.24 | 4473.9 |

Table 1: Experiments: (1-9) Loops not enforced, (10-18) Loops enforced

in ASP is flexible and extensible. Note that extensions like constraints enforcing separation and binding of duties [10] can be easily added in our formalism, which we omitted due to space restrictions.

We plan to conduct further performance measurements and compare them to other formalisms, e.g., constraint solvers. We are confident that there is room for optimisations (e.g., symmetry breaking [5] or similar techniques) that have been successfully applied in ASP.

## References


[1] Gerhard Brewka, Thomas Eiter, and Mirosław Truszczyński. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2011.

[2] Francesco Calimeri, Martin Gebser, Marco Maratea, and Francesco Ricca. The Design of the Fifth Answer Set Programming Competition. *CoRR*, 2014.

[3] J. Carlier and E. Pinson. An Algorithm for Solving the Job-shop Problem. *Manage. Sci.*, 35(2):164–176, February 1989.

[4] Yann Chevaleyre, Paul E. Dunne, Ulle Endriss, Jrme Lang, Michel Lematre, Nicolas Maudet, Julian Padget, Steve Phelps, Juan A. Rodrguez-aguilar, and Paulo Sousa. Issues in multiagent resource allocation. *Informatica*, 30:2006, 2006.

[5] Andreas A. Falkner, Gottfried Schenner, Gerhard Friedrich, and Anna Ryabokon. Testing object-oriented configurators with ASP. In *Workshop on Configuration at ECAI 2012*, pages 21–26, 2012.

[6] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

[7] Stefan Heinz and Christopher Beck. Solving Resource Allocation/Scheduling Problems with Constraint Integer Programming. In *COPLAS 2011*, pages 23–30, 2011.

[8] M.H.A. Hendriks, B. Voeten, and L. Kroep. Human resource allocation in a multi-project R&D environment: Resource capacity allocation and project portfolio planning in practice. *Int. J. of Project Management*, 17(3):181–188, 1999.

[9] Marijn JH Heule and Torsten Schaub. What's Hot in the SAT and ASP Competitions. In *AAAI*, 2015.

[10] Maria Leitner and Stefanie Rinderle-Ma. A systematic review on security in Process-Aware Information Systems Constitution, challenges, and future directions. *Information and Software Technology*, 56(3):273 – 293, 2014.

[11] Niels Lohmann, Eric Verbeek, and Remco Dijkman. Petri Net Transformations for Business Processes - A Survey. *Transactions on Petri Nets and Other Models of Concurrency II*, 2:46–63, 2009.

[12] Tadao Murata. Petri nets: Properties, analysis and applications. *IEEE*, 77(4):541–580, 1989.

[13] Louchka Popova-Zeugmann. Time Petri Nets. In *Time and Petri Nets*, pages 139–140. Springer Berlin Heidelberg, 2013.

[14] A. Rozinat and R. S. Mans. Mining CPN Models: Discovering Process Models with Data from Event Logs. In *Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN*, pages 57–76, 2006.

[15] Pinar Senkul and Ismail H. Toroslu. An Architecture for Workflow Scheduling Under Resource Allocation Constraints. *Inf. Syst.*, 30(5):399–422, July 2005.

[16] W.M.P. van der Aalst. Petri net based scheduling. *Operations-Research-Spektrum*, 18(4):219–229, 1996.

[17] Jan Weglarz. Project Scheduling with Continuously-Divisible, Doubly Constrained Resources. *Management Science*, 27(9):1040–1053, 1981.

[18] Yuhong Yan, Z. Maamar, and Weiming Shen. Integration of workflow and agent technology for business process management. In *Computer Supported Cooperative Work in Design*, pages 420–426, 2001.

[19] Chongjie Zhang, Victor Lesser, and Prashant Shenoy. A Multi-Agent Learning Approach to Online Distributed Resource Allocation. In *International Joint Conference on Artificial Intelligence (IJCAI'09)*, volume 1, pages 361–366, 2009.

Published in the Proceedings of the 14th Int. Conference on Business Process Management: Business Process Management Forum (BPM Forum 2016), pp. 3-19, Sep 2016, Springer LNBIP vol. 260

# Resource Allocation with Dependencies in Business Process Management Systems*

Giray Havur, Cristina Cabanillas, Jan Mendling, and Axel Polleres

Vienna University of Economics and Business, Austria
{firstname.lastname}@wu.ac.at

**Abstract.** Business Process Management Systems (BPMS) facilitate the execution of business processes by coordinating all involved resources. Traditional BPMS assume that these resources are *independent* from one another, which justifies a greedy allocation strategy of offering each work item as soon as it becomes available. In this paper, we develop a formal technique to derive an optimal schedule for work items that have *dependencies* and *resource conflicts*. We build our work on Answer Set Programming (ASP), which is supported by a wide range of efficient solvers. We apply our technique in an industry scenario and evaluate its effectiveness. In this way, we contribute an explicit notion of resource dependencies within BPMS research and a technique to derive optimal schedules.

**Keywords:** Answer Set Programming, optimality, resource allocation, resource requirements, work scheduling

## 1 Introduction

Business Process Management Systems (BPMS) have been designed as an integral part of the business process management (BPM) lifecycle by coordinating all resources involved in a process including people, machines and systems [26]. At design time, BPMS take as input a business process model enriched with technical details such as role assignments, data processing and system interfaces as a specification for the execution of various process instances. In this way, they support the efficient and effective execution of business processes [21].

It is an implicit assumption of BPMS that work items are *independent* from one another. If this assumption holds, it is fine to put work items in a queue and offer them to available resources right away. This approach of resource allocation, can be summarized as a greedy strategy. However, if there are *dependencies* between work items, this strategy can easily become suboptimal. Some domains like engineering or healthcare have a rich set of activities for which various resources, human and non-human, are required at the same time. Resource conflicts have often the consequence that working on one work item blocks resources such that other work items cannot be worked on. This observation emphasizes the need for techniques to make better use of existing resources in business processes [25].

* Funded by the Austrian Research Promotion Agency (FFG) grant 845638 (SHAPE).

In this paper, we address current limitations of BPMS with respect to taking such resource constraints into account. We extend prior research on the integration of BPMS with calendars [16] to take dependencies and resource conflicts between work items into account. We develop a technique for specifying these dependencies in a formal way in order to derive a globally optimal schedule for all resources together. We define our technique using Answer Set Programming (ASP), a formalism from logic programming that has been found to scale well for solving problems as the one we tackle [13]. We evaluate our technique using an industry scenario from the railway engineering domain. Our contribution to research on BPMS is an explicit notion of dependence along with a technique to achieve an optimal schedule.

The paper is structured as follows. Section 2 presents and analyzes an industry scenario. Section 3 conceptually describes the resource allocation problem. Section 4 explains our ASP-based solution and how it can be applied to the industry scenario. Section 5 evaluates the solution. Section 6 discusses related work. Section 7 summarizes the conclusions of the work and the future steps.

## 2 Motivation

In the following, we describe an industry scenario that leads us to a more detailed definition of the resource allocation problem and its complexity.

### 2.1 Industry Scenario

A company that provides large-scale technical infrastructure for railway automation requires rigorous testing for the systems deployed. Each system consists of different types and number of hardware that are first set up in a laboratory. This setup is executed by some employees specialized in different types of hardware. Afterwards, the simulation is run under supervision.

Figure 1 depicts two process models representing the setup and run phases of two tests. We use (timed) Petri nets [20] for representing the processes. The process activities are represented by transitions ($a_i$). The number within square brackets next to the activities indicates their (default maximum) duration in generic time units (TU). The numbers under process names indicate the starting times of the process executions: 8 TU for Test-1 and 12 TU for Test-2. The processes are similar for all the testing projects but differ in the activities required for setting up the hardware as well as in the resource requirements associated with them. Certain resources can only be allocated to activities during working periods, i.e., we want to enforce time intervals (so called *breaks*) where some resources are not available. In our scenario, no resource is available in the intervals $[0, 8)$,$[19, 32)$,$[43, 56)$, and $[67, 80)$.

For completing tests, the available non-human resources in the organization include 13 units of space distributed into 2 laboratories (Table 1) and several units of 3 types of hardware (Table 2). The human resources of the company are specialized in the execution of specific phases of the two testing projects, whose activities are able to complete in a specific time. Table 3 shows available resources in different process phases and

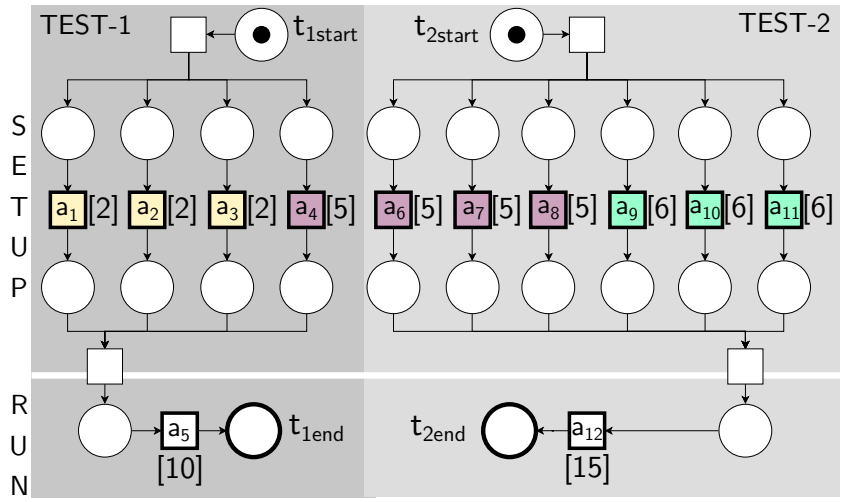



Fig. 1: Workflow for two projects

therefore, their ability to conduct certain activities along with their years of experience in the company in square brackets.

The requirements on the use of such resources in the process activities are shown in Table 4. Each process activity requires a specific set of resources for its completion. For instance, three of the activities involved in the setup of Test-1 require 1 employee working on 1 unit of the hardware HW-1 in a laboratory; 1 setup activity requires 1 employee working on 1 unit of the hardware HW-2 in a laboratory; and the run activity requires 4 employees. Besides, a test can only be executed if the whole setup takes place in the same laboratory.

The aim in this scenario is to optimize the overall execution time of simultaneous tests and consequently, the space usage in the laboratories.

| | $LAB-1$ | $LAB-2$ |
|---|---|---|
| Space | 4 | 9 |

Table 1: Available space in labs

| $Type$ | $Units$ |
|---|---|
| $HW1$ | hw1a, hw1b, hw1c |
| $HW2$ | hw2a, hw2b, hw2c, hw2d |
| $HW3$ | hw3a, hw3b, hw3c |

Table 2: Available hardware (HW)

| | $Test-1$ | | $Test-2$ | |
|---|---|---|---|---|
| | $Setup$ | $Run$ | $Setup$ | $Run$ |
| Glen[7] | ✓ | ✓ | | |
| Drew[7] | | ✓ | | |
| Evan[3] | | ✓ | | |
| Mary[5] | | ✓ | ✓ | |
| Kate[6] | | | ✓ | ✓ |
| Amy[8] | ✓ | | ✓ | ✓ |

Table 3: Specialization of employees

### 2.2 Insights

The resource allocation problem[1] deals with the assignment of resources and time intervals to the execution of activities. The complexity of resource allocation in BPM arises from coordinating the explicit and implicit dependencies across a broad set of resources and activities of processes as well as from solving potential conflicts on the use of certain resources. As we observe in our industry scenario, such dependencies include, among others: (i) resource requirements, i.e., the characteristics of the resources that are involved in an activity (e.g., roles or skills) (cf. Table 3); (ii) temporal requirements. For instance, the duration of the activities may be static or may depend on the characteristics of the set of resources involved in it, especially for collaborative activities in which several employees work together (such as for the activities of the run phase of a testing process). Furthermore, resource availability may not be unlimited (e.g., break calendars). In addition, resource conflicts may emerge from interdependencies between requirements, e.g., activities might need to be executed within a specific setting which may be associated with (or share resources with) the setting of other activities (e.g., all the setup activities of a testing process must be performed in the same laboratory).

A resource allocation is *feasible* if (1) activities are scheduled with respect to time constraints derived from activity durations and control flow of the process model, and (2) resources are allocated to scheduled activities in accordance with resource availability and resource requirements of activities. This combinatorial problem for finding a feasible resource allocation under constraints is an *NP-Complete* problem [27]. However, organizations generally pursue an optimal allocation of resources to process activities aiming at minimizing overall execution times or costs, or maximizing the usage of the resources available. In presence of objective functions the resource allocation problem becomes $\Delta_2^P$ [5].

## 3 Conceptualization of the Resource Allocation Problem

Fig. 2 illustrates our conceptualization of the resource allocation problem. We divide it into three complexity layers related to the aforementioned dependencies and resource

[1] Commonly referred as *scheduling*.

| | *Activities* | *Requirements* |
|---|---|---|
| *Test-1* | $a_1 - a_3$ | 1 *Employee*:Setup-1, 1 *Hardware*:HW-1, 1 *Lab*:$a_1$-$a_4$ same lab |
| | $a_4$ | 1 *Employee*:Setup-1, 1 *Hardware*:HW-2, 1 *Lab*:$a_1$-$a_4$ same lab |
| | $a_5$ | 4 *Employee*:Run-1, after execution(a.e.) release the lab for $a_1$-$a_4$ |
| *Test-2* | $a_6 - a_8$ | 1 *Employee*:Setup-2, 1 *Hardware*:HW-2, 1 *Lab*:$a_6$-$a_{11}$ same lab |
| | $a_9 - a_{11}$ | 1 *Employee*:Setup-2, 1 *Hardware*:HW-3, 1 *Lab*:$a_6$-$a_{11}$ same lab |
| | $a_{12}$ | 2 *Employee*:Run-2 (hasExp>5), a.e. release the lab for $a_6$-$a_{11}$ |

Table 4: Activity requirements

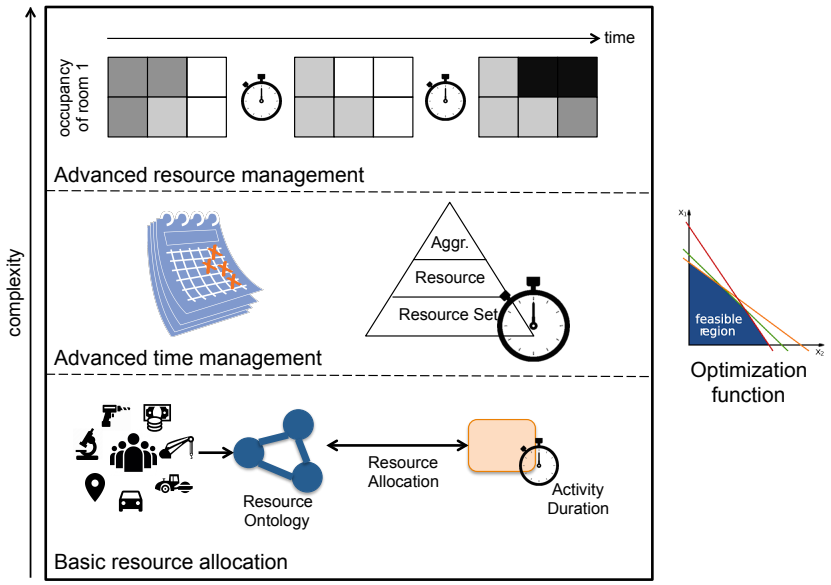


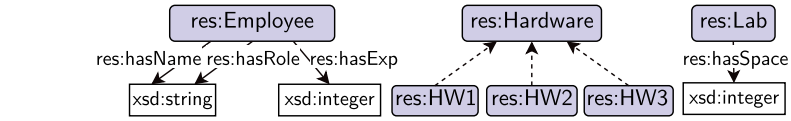

Fig. 2: Resource allocation in business processes

conflicts. Optimization functions can be applied to all types of allocation problems. This model has been defined from the characteristics identified in our industry scenario as well as in related literature [19].

### 3.1 Basic Resource Allocation

Three elements are involved in a basic resource allocation, namely: a model that stores all the information required about the resources available, information about the expected duration of the process activities, and a language for defining the restrictions that characterize the allocation.

**Resource Ontology** As a uniform and standardized representation language, we suggest the use of RDF Schema (RDFS) [4] to model organizational information and resources. Fig. 3 illustrates a sample RDFS ontology, in which a *resource* is characterized by a *type* and can have one or more *attributes*. In particular, any resource type (e.g. *Employee*) is a subclass of rdfs:Resource. The attributes are all of type rdf:Property; domain (rdfs:domain) and range of attributes are indicated with straight arrows labeled with the attribute name, whereas dashed arrows indicate an rdfs:subclassOf. There are three different types of resources: *Employee*, *Hardware* and *Lab*, where *Hardware* has three resource subtypes. Employees have attributes for their name (*hasName*), role(s) (*hasRole*) and experience level (*hasExp*) in the organization (number of years). Labs provide a certain amount of space for experiments (*hasSpace*). An instantiation of the ontology is described at the bottom of the figure using the RDF Turtle syntax [2]. This instantiation represents Tables 1-3 of the industry scenario.




```
:glen a res:Employee; res:name "Glen"; res:hasExp 7;
       res:hasRole "Setup_Test1", "Run_Test1" .
:drew a res:Employee; res:name "Drew";
       res:hasExp 7; res:hasRole "Run_Test1" .
 ...
:lab1 a res:Lab, res:hasSpace 4 .
:lab2 a res:Lab, res:hasSpace 9 .
:hw1a a res:HW1. :hw1b a res:HW1. :hw1c a res:HW1.
:hw2a a res:HW1. :hw2b a res:HW2. :hw2c a res:HW2. :hw2d a res:HW2.
:hw3a a res:HW3. :hw3b a res:HW3. :hw3c a res:HW3.
```

Fig. 3: Resource ontology and example instantiation

**Activity Duration** Resource allocation aims at properly distributing available resources among running and coming work items. The main temporal aspect is determined by the expected duration of the activities. The duration can be predefined according to the type of activity or calculated from previous executions, usually taking the average duration as reference. This information can be included in the executable process model as a property of an activity (e.g. with BPMN [18]) or can be modelled externally. In either case, it has to be accessible by the allocation algorithm.

**Resource Allocation** Resource allocation can be seen as a two-step definition of restrictions. First, the so-called *resource assignments* must be defined, i.e., the restrictions that determine which resources can be involved in the activities [6] according to their properties. The outcome of resource assignment is one or more[2] *resource sets* with the set of resources that can be potentially allocated to an activity at run time. The second step assigns cardinality to the resource sets such that different settings can be described, e.g. for the execution of activity $a_1$, 1 employee with role *setup-1*, 1 hardware of type *HW2*, and 1 unit space of a laboratory are required.

There exist languages for assigning resource sets to process activities [6, 33, 32, 7]. However, cardinality is generally disregarded under the assumption that only one resource will be allocated to each process activity. This is a limitation of current BPMS that prevents the implementation of industry scenarios like the one described in Section 2.1.

### 3.2 Advanced Time Management

This layer extends the temporal aspect of resource allocation by taking into account that: (i) resource availability affects allocation, and that (ii) the resource sets allocated to an activity may affect its duration. Regarding resource availability, calendars are

[2] Since several sets of restrictions can be provided, e.g. for activity $a_1$ resources with either role $r_1$ or skill $s_1$ are required.

an effective way of specifying different resource availability status, such as available, unavailable, occupied/busy or blocked [19]. Such information must be accessible by the resource allocation module. As for the variable activity durations depending of the resource allocation, three specificity levels can be distinguished:

- *Resource-set-based duration*, i.e., a triple $(activity, resourceSet, duration)$ stating the (minimum/average) amount of time that it takes to the resources within a specific resource set (i.e., cardinality is disregarded) to execute instances of a certain activity. For instance, $(a_1, technician, 6)$ specifies that people with the role $technician$ need (at least/on average) 6 TU to complete activity $a_1$, assuming that $technician$ is an organisational role.
- *Resource-based duration*, i.e., a triple $(activity, resource, duration)$ stating the (minimum/average) amount of time that it takes to a concrete resource to execute instances of a certain activity. For instance, $(a_1, John, 8)$ specifies that $John$ needs (at least/on average) 8 TU to complete activity $a_1$.
- *Aggregation-based duration*, i.e., a triple $(activity, group, duration)$ stating the (minimum/average) amount of time that it takes to a specific group to execute instances of a certain activity. In this paper, we use $group$ to refer to a set of human resources that work together in the completion of a work item, i.e., cardinality is considered. Therefore, a $group$ might be composed of resources from different resource sets which may not necessarily share a specific resource-set-based duration. An aggregation function must be implemented in order to derive the most appropriate duration for an activity when a group is allocated to it. The definition of that function is up to the organization. For instance, a group might be composed of $(John, Claire)$, where $John$ has an associated duration of 8 TU for activity $a_1$ and $Claire$ does not have a specific duration but she has role $technician$, with an associated duration of 6 TU for activity $a_1$. Strategies for allocating the group to the activity could be to consider the maximum time needed for the resources involved (i.e., 8 TU), or to consider the mean of all the durations (i.e., 7 TU) assuming that the joint work of two people will be faster than one single resource completing all the work.

### 3.3 Advanced Resource Management

The basic resource allocation layer considers resources to be *discrete*, i.e. they are either fully available or fully busy/occupied. This applies to many types of resources, e.g. people, software or hardware. However, for certain types of non-human resources, availability can be partial at a specific point in time. Moreover, they may have other *fluent* attributes. For instance, *cumulative resources* are hence characterized by their *dynamic* attributes and they can be allocated to more than one activity at a time, e.g. in Fig. 2 there is a resource *room 1* whose occupancy changes over time.

We use the ASP solver $clasp$ [12] due to its efficiency for our experiments. This allows us to use integer variables as attributes. There are also other extensions of ASP such as FASP [35] that adds the power to model continuous variables.

### 3.4 Optimization Function

Searching for (the existence of) a feasible resource allocation ensures that all the work items can eventually be completed with the available resources. However, typically schedules should also fulfill some kind of optimality criterion, most commonly completion of the schedule in the shortest possible overall time. Other optimization criteria may involve for instance costs of the allocation of certain resources to particular activities, etc.

Given such an optimization criterion, there are greedy approaches [34] providing a substantial improvements over choosing any feasible schedule, although such techniques depend on heuristics and may not find a globally optimal solution for complex allocation problems.

We refer to [24] for further information on various optimization functions, but emphasize that our approach will in principle allow arbitrary optimization functions and finds optimal solutions – similar in spirit to encodings of cost optimal planning using ASP [10].

## 4 Implementation with ASP

Answer Set Programming (ASP) [12] is a declarative (logic-programming-style) paradigm. Its expressive representation language, ease of use, and computational effectiveness facilitate the implementation of combinatorial search and optimization problems (primarily *NP-hard*). Modifying, refining, and extending an ASP program is uncomplicated due to its strong declarative aspect.

An *ASP program* $\Pi$ is a finite set of rules of the form:

$$A_0 \leftarrow A_1, \ldots, A_m, \mathit{not}\ A_{m+1}, \ldots, \mathit{not}\ A_n. \tag{1}$$

where $n \geq m \geq 0$ and each $A_i \in \sigma$ are (function-free first-order) atoms; if $A_0$ is empty in a rule $r$, we call $r$ a constraint, and if $n = m = 0$ we call $r$ a fact.

Whenever $A_i$ is a first-order predicate with variables within a rule of the form (1), this rule is considered as a shortcut for its *grounding* $ground(r)$, i.e., the set of its ground instantiations obtained by replacing the variables with all possible constants occurring in $\Pi$. Likewise, we denote by $ground(\Pi)$ the set of rules obtained from grounding all rules in $\Pi$. Sets of rules are evaluated in ASP under the so-called stable-model semantics, which allows several models, so called *answer sets* (cf. [3] for details).

ASP Solvers typically first compute a subset of $ground(\Pi)$ and then use a DPLL-like branch and bound algorithm to find answer sets for this ground program. We use the ASP solver $clasp$ [12] for our experiments as it has proved to be one of the most efficient implementations available [8].

As syntactic extension, in place of atoms, $clasp$ allows set-like *choice expressions* of the form $E = \{A_1, \ldots, A_k\}$ which are true for any subset of $E$; that is, when used in heads of rules, $E$ generates many answer sets, and such rules are often referred to as *choice rules*. Another extension supported in $clasp$ are optimization statements [12] to indicate preferences between possible answer sets:

$$\#\mathit{minimize}\ \{A_1 : Body_1 = w_1, \ldots, A_m : Body_m = w_m @ p\}$$

associates integer weights (defaulting to 1) with atoms $A_i$ (conditional to $Body_i$ being true), where such a statement expresses that we want to find only answer sets with the smallest aggregated weight sum; again, variables in $A_i : Body_i = w_i$ are replaced at grounding w.r.t. all possible instantiations. Several optimization statements can be introduced by assigning the statement a priority level $p$. Reasoning problems including such weak constraints are $\Delta_2^P$-complete.

Finally, many problems conventiently modelled in ASP require a boundary parameter `k` that reflects the size of the solution. However, often in problems like planning or model checking this boundary (e.g. the plan length) is not known upfront, and therefore such problems are addressed by considering one problem instance after another while gradually increasing this parameter `k`. Re-processing repeatedly the entire problem is a redundant approach, which is why incremental ASP (iASP) [12] natively supports incremental computation of answer sets; the intuition is rooted in treating programs in program slices (extensions). In each incremental step, a successive extension of the program is considered where previous computations are re-used as far as possible.

A former version of our technique is detailed in [13]. We enhance our encoding in three folds: (1) basic resource allocation supporting multiple business processes with multiple running instances, (2) definition of *advanced resource management* concepts, and (3) definition of *advanced time management* concepts. The entire ASP encoding can be found at `http://goo.gl/Q7B2t4`.

### 4.1 Basic Resource Allocation

This program schedules the activities in business processes described as timed Petri nets (cf. the generic formulation of 1-safe Petri Nets [13, Section 4]) and allocates resources to activities with respect to activity-resource requirements. To achieve this, the program finds a firing sequence between initial and goal places of given processes, schedules the activities in between, and allocates resources by complying with resource requirements. In our program, a firing sequence is represented as predicates `fire(a,b,i,k)`, which means that an activity `a` of a business process `b` in instance `i` is fired at step `k`. Starting time of each activity in the firing sequence is derived from the time value accumulated at the activity's input place `p`. A time value at a place `p` is represented by the predicate `timeAt(p,c,b,i,k)`, where `c` is the time value.

A *resource set* is defined as a rule that derives the members of the set that satisfy a number of properties. These properties can be class memberships or resource attributes defined in resource ontology(cf. Section 3.1). Note that, any resource ontology described in RDF(S) can be easily incorporated/translated into ASP [11]. A resource set is represented with the predicate `resourceSet(R,id)`, where `R` is a set of discrete resources and `id` is the identifier of the set. We explain the following resource sets following our industry scenario:

*All employees that can take part in the setup phase of Test-1:*

```
resourceSet(R,rs_set1):-employee(R), hasRole(R,setup1).
```

*All employees that can take part in the run phase of Test-2 and have a working experience greater than 5 years:*

```
resourceSet(R,rs_ex2):-employee(R), hasRole(R,run2), hasExp(E),
E>5.
```

*All hardware resources of type HW2:*
`resourceSet(R,rs_h2):-hardware2(R).`

After defining resource sets, we define *resource requirements* of an activity `a` with the predicate `requirement(a,id,`$n$`)` where `id` refers to a specific resource set and $n$ is the number of resources that activity `a` requires from this set. For instance, `requirement(`$\mathtt{a}_{12}$`,rs_ex2,2)` means that activity $\mathtt{a}_{12}$ requires 2 resources from the resource set `rs_ex2`. The resource requirements that we support include typical access-control constraints [6]. In particular, *Separation of duties (SoD)* and *binding of duties(BoD)* are implemented in our program by using the predicate `separateDuties(`$\mathtt{a}_1,\mathtt{b}_1,\mathtt{a}_2,\mathtt{b}_2$`)`, which separates the resources allocated to the activity $\mathtt{a}_1$ of process $\mathtt{b}_1$ from the resources allocated to $\mathtt{a}_2$ of $\mathtt{b}_2$; and `bindDuties(`$\mathtt{a}_1,\mathtt{b}_1,\mathtt{a}_2,\mathtt{b}_2$`)`, which binds the resources allocated to the activity $\mathtt{a}_1$ of process $\mathtt{b}_1$ with the resources allocated to $\mathtt{a}_2$ of $\mathtt{b}_2$.

### 4.2 Advanced Time Management

Default durations of activities are defined in the timed Petri nets and represented as `activityDuration(T,D)` in our program. This default duration can be overwritten by `d` when any resource `r` that belongs to a resource set `rs` is assigned to a certain activity `a` of the process `b` by using the predicate `rSetActDuration(rs,a,b,d)`. In a similar fashion, the default duration can be overwritten by a new value `d` when a certain resource `r` is assigned to a certain activity `a` of the process `b` by using the predicate `resActDuration(r,a,b,d)`. The order (>) preferred in activity time is `resActDuration>rSetActDuration>activityDuration`. This is especially useful when a resource or a resource set is known to execute a particular activity in a particular amount of time, which can be different from the default duration of the activity.

As one activity can be allocated to a group of resources (cf. Section 3.2), an aggregation method might be needed. Our default aggregation method identifies the maximum duration within the group and uses it for allocation. This method can be modified with different aggregation options that fit in the purpose of allocation scenario.

In many real-life projects, certain resources are only available during the working periods (a.k.a. *break calendars*). We model this by `break(rs,` $\mathtt{c}_1,\mathtt{c}_2$`)` that forbids allocation of resources in the resource set `rs` between time $\mathtt{c}_1$ and $\mathtt{c}_2$, where $\mathtt{c}_1 < \mathtt{c}_2$.

For business process instances and their activities, (optionally, max. or min.) starting or ending times can be defined using the following predicates:
`actStarts(o,a,b,i,c)`, i.e. activity `a` in business process `b` of instance `i`, starts <o> at `c`; `actEnds(o,a,b,i,c)`, i.e. activity `a` in business process `b` of instance `i`, ends <o> at `c`; `bpiStarts(o,b,i,c)`, i.e. business process `b` of instance `i`, starts <o> at `c`; `bpiEnds(o,b,i,c)`, i.e. business process `b` of instance `i`, ends <o> at `c`; where `o`$\in$ {`strictly,earliest,latest`}.

### 4.3 Advanced Resource Management

A cumulative resource has an integer value attribute describing the state of the resource. This value can increase or decrease when the resource is *consumed* or *generated* by an

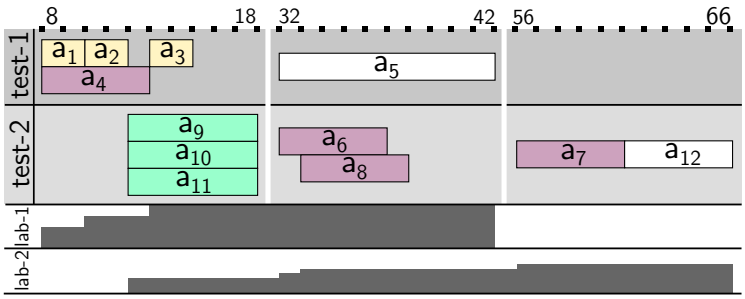



Fig. 4: Optimal resource allocation for our industry scenario

activity requiring it. Definition of cumulative resource sets have one extra term for this reason: `resourceSet(R,V,id)`, where `R` is the set of cumulative resources, `V` is the set of their initial value and `id` is the identifier of the resource set. For example:
*Lab space set:*

```
resourceSet(R,V,lab_space):-lab(R),hasSpace(R,V).
```

Resource requirements are defined like for discrete resources, where $n$ is the amount of resource consumed or generated. For instance, `requirement(a1,lab_space,1)` consumes 1 unit of lab space when $a_1$ is allocated, whereas `requirement(a12,lab_space,-6)` releases 6 units of space by the time $a_{12}$ is completed.

*Resource blocking* functionality allows us to block some resources between the execution of two activities in a process. A blocked resource is not allowed to be allocated by an activity in this period. `block(a1,a2,id,n)` blocks `n` amount of resources in the resource set `id` from the beginning of $a_1$ to beginning of $a_2$.

### 4.4 Optimization Function

As aforementioned, the ASP solver *clasp* allows defining objectives as cost functions that are expressed through a sequence of `#minimize` statements. In our encoding, we ensure time optimality of our solutions using a minimization statement. The incremental solver finds an upper-bound time value $c_{upper}$ at step `k`. A time optimal solution is guaranteed at step `k'` where `k'` $= c_{upper}/min($`D`$)$, `D` is the set of activity durations. In a similar way, any objective that is quantified with an integer value (e.g. cost objectives, resource leveling, etc.) could be introduced. When there is more than one objective, they should be prioritized.

Taking into account all the aforementioned functionality, using the encoding summarized above and detailed in `http://goo.gl/Q7B2t4`, a time optimal solution for our industry scenario is depicted in Fig. 4. The final allocation of resources to each activity $a_i$ is as follows:

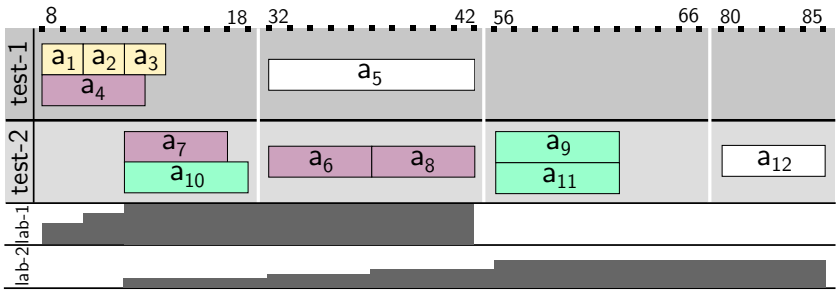



Fig. 5: A greedy (suboptimal) resource allocation for our industry scenario

$a_1$ {Amy,hw1a,lab-1(1)}
$a_2$ {Amy,hw1b,lab-1(1)}
$a_3$ {Glen,hw1c,lab-1(1)}
$a_4$ {Glen,hw2b,lab-1(1)}
$a_5$ {Glen,Drew,Ewan,Mary,lab-1(-4)}
$a_6$ {Kate,hw2c,lab-2(1)}
$a_7$ {Mary,hw2a,lab-2(1)}
$a_8$ {Amy,hw2d,lab-2(1)}
$a_9$ {Amy,hw3c,lab-2(1)}
$a_{10}$ {Mary,hw3b,lab-2(1)}
$a_{11}$ {Kate,hw3a,lab-2(1)}
$a_{12}$ {Kate,Amy,lab-2(-6)}

## 5 Evaluation

Our resource allocation technique not only finds an optimal schedule for activities in our industry scenario but also consequently optimizes the resource utilization. We show the improvement in result quality by comparing an optimal allocation of the scenario (cf. Fig 4) against a greedy allocation, depicted in Fig. 5. We use the following two criteria for this comparison:
*1. Total execution time (TET)* corresponds to the end time of the last activity for each process (e.g. $a_5$ for process Test-1).
*2. Average employee utilization (AEU):* For any time unit $c \in C$, $c_{start}$ is the start time, $c_{end}$ is the end time of process execution, $c_{start} \leq c \leq c_{end}$, a function $s : c \rightarrow R_b$ returns an ordered set of billable employees $R_b$ respecting Table 3. For each element $s \in R_b$ a function $w_c : r \rightarrow \{0, 1\}$ returns whether the employee $r$ is working at time $c$. In other words, we first sum the ratio between the number of employees allocated and the total number of employees that potentially can take part at each time unit, and normalize this sum using the overall execution time. $AEU$ is calculated as described by (2).

$$AEU = \frac{\sum_{i=c_{start}}^{c_{end}} \frac{\sum_{r \in s(i)} w_c(r)}{|s(i)|}}{c_{end} - c_{start}} \tag{2}$$

For instance, in Fig. 5, $s(8) = \{Glen, Drew, Evan, Mary, Amy\}$. Note that $Kate$ is not in the set since she only takes part in Test-2 and Test-2 instances have not started due to the deadline constraint `bpStarts(earliest,test-2,12)`. At time 8, only $w_c(Amy)$ and $w_c(Glen)$ have value of 1.

Table 5 summarizes the results obtained using the two aforementioned criteria for the two allocation strategies. The execution of our industry scenario finishes 5 TU before under optimal allocation, which corresponds to 14% of time usage improvement

| | Optimal(Fig. 4) | Greedy (Fig. 5) |
|---|---|---|
| $TET$ | 30 | 35 |
| $AEU$ | 0.61 | 0.54 |

Table 5: Result quality comparison

| | Basic Resource Allocation | | Advanced Time Management | | Advanced Res. Mgmt. | | |
|---|---|---|---|---|---|---|---|
| Approach | Res. Type | A. Level | Calendar | Aggreg. | Dynamism | Objective | Formalism |
| [9] | Both | Low | ✓ | - | - | Usage | MIP |
| [29] | Both | Medium | ✓ | - | - | Usage | IP |
| [23] | Both | High | ✓ | - | - | Any | Ad-hoc |
| [30] | Both | Medium | - | ✓ | - | Time&usage | LIP |
| [17] | Both | Medium | - | ✓ | - | Time&usage | CP |
| [31] | Both | Medium | - | ✓ | - | Makespan | Ad-hoc |
| [28] | Both | Medium | - | ✓ | - | - | CP |
| [34] | Both | Medium | - | ✓ | - | Makespan | Petri N. |
| [13] | Human | Medium | - | ✓ | - | Time | ASP |

Table 6: Representative approaches related to resource allocation

while $AEU$ improves 7%. We refer the reader to [13] for scalability of our technique, where we demonstrated that ASP performs well for resource allocation in the BPM domain.

## 6 Related Work

Resource allocation has been extensively explored in various domains for addressing everyday problems, such as room, surgery or patient scheduling in hospitals, crew-job allocation or resource leveling in organizations. Table 6 collects a set of recent, representative approaches of three related domains: operating room scheduling [9, 29, 23], project scheduling [30, 17, 31] and resource allocation in business processes [34, 28, 13]. The features described in Section 3 are used for comparing them[3]. Specifically, column *Res. Type* specifies the type(s) of resource(s) considered for allocation (human, non-human or both); column *A. Level* indicates the expressiveness of the restrictions that can be defined for the allocation, among: (i) low, when a small range of resource assignment requirements are considered *and* only one individual of each resource type (e.g., one person and one room) is allocated to an activity, i.e., cardinality is disregarded; (ii) medium, when a small range of resource assignment requirements are considered *or* cardinality is disregarded; and (iii) high, when flexible resource assignment *and* cardinality are supported; column *Calendar* refers to whether information about resource availability is taken into account (a blank means it is not); column *Aggreg.* indicates whether the execution time of an activity is determined by the resources involved in it; column *Advanced Res. Mgmt.* shows the support for cumulative resources

[3] We have adopted the vocabulary used in BPM for resource allocation [34, 28].

that can be shared among several activities at the same time; column *Objective* defines the variable to be optimized; and column *Formalism* specifies the method used for resolving the problem.

The concept of process is not explicitly mentioned in the operating room scheduling problem. Traditional approaches in this field tended to adopt a two-step approach which, despite reducing the problem complexity, failed to ensure optimal or even feasible solutions [23]. It is a property of the surgery scheduling problem that some resources, such as the operating rooms, can only be used in one project at a time [23], so cardinality is disregarded [9, 29]. However, it is important to take into account resource availability. The most expressive approach in this domain [23] is an ad-hoc algorithm, whereas integer programming (IP) stands out as a formalism to efficiently address this problem.

Project scheduling consists of assigning resources to a set of activities that compose a project, so the concept of workflow is implicit. The approaches in this domain support cardinality for resource allocation but they rely on only the resource type for creating the resource sets assigned to an activity. These approaches implement the so-called *resource-time tradeoff*, which assumes that activity completion is faster if two resources of the same type work together in its execution [30, 17] (cf. Section 3.2). However, they assume a constant per-period availability of the resources [31], hence calendars are overlooked. The project scheduling problem has been repeatedly addressed with formalisms like linear integer programming (LIP) [30] and constraint programming (CP) [17], yet ad-hoc solutions also exist [31].

Finally, in the domain of BPM, the state of the art in resource allocation does not reach the maturity level of the other domains despite the acknowledged importance of the problem [1] and the actual needs (cf. Section 2.1). Similar to project scheduling, a constant availability of resources is typically assumed. In addition, due to the computational cost associated to joint resource assignment and scheduling problems [15], the existing techniques tend to search either for a feasible solution without applying any optimizations [28]; or for a local optimal at each process step using a greedy approach that might find a feasible but not necessarily a globally optimal solution [34]. Nonetheless, recently it was shown that global optimization is possible at a reasonable computational cost [13]. Moreover, driven by the limitations of current BPMS, which tend to disregard collaborative work for task completion, cardinality has been unconsidered for allocation, giving rise to less realistic solutions.

In general, the optimization function depends on the problem and the objective of the approach but it is generally based on minimizing time, makespan or cost, or making an optimal use of the resources (a.k.a. resource leveling [22]).

## 7 Conclusions and Future Work

In this paper we have conceptualized the complex problem of resource allocation under realistic dependencies that affect resources and activities as well as potential conflicts that may arise due to simultaneous requirement of resources. Our implementation based on ASP and its evaluation show that optimal solutions for this problem are possible, which extends the state of the art in BPM research and could contribute to extend the support in existing BPMS. ASP has proved to scale well [8] and can be easily integrated with RDF ontologies [11].

It is not the aim of this work to provide an end-user-oriented but an effective solution. In order to reasonably use our ASP implementation with a BPMS, it is required: (i) to map the notation used for process modeling along with the durations associated with the activities to (timed) Petri nets, for which several techniques have been designed [14]; and (ii) the integration of languages for defining all the requirements which could be used by non-technical users in the system as well as their mapping to ASP. However, to the best of our knowledge, there is not yet such an expressive end-user-oriented language but languages that allow a partial definition of the requirements [7, 33].

As future work we plan to compare our technique with existing approaches on other optimal resource allocation techniques, explore the preemptive resource allocation as well as to apply our technique in other domains.

## References


[1] Michael Arias, Eric Rojas, Jorge Munoz-Gama, and Marcos Sepúlveda. A Framework for Recommending Resource Allocation based on Process Mining. In *BPM 2015 Workshops (DeMiMoP)*, page In press, 2015.

[2] David Beckett, Tim Berners-Lee, Eric Prud'hommeaux, and Gavin Carothers. Turtle – Terse RDF Triple Language. W3C Candidate Recommendation, February 2014. `https://www.w3.org/TR/turtle/`.

[3] Gerhard Brewka, Thomas Eiter, and Mirosław Truszczyński. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2011.

[4] Dan Brickley and R.V. Guha. RDF Schema 1.1. W3C Recommendation, February 2014. `http://www.w3.org/TR/rdf-schema/`.

[5] Francesco Buccafurri, Nicola Leone, and Pasquale Rullo. Enhancing disjunctive datalog by constraints. *IEEE Trans. on Knowledge and Data Engineering*, 12(5):845–860, 2000.

[6] Cristina Cabanillas, Manuel Resinas, Adela del Río-Ortega, and Antonio Ruiz-Cortés. Specification and Automated Design-Time Analysis of the Business Process Human Resource Perspective. *Inf. Syst.*, 52:55–82, 2015.

[7] Cristina Cabanillas, Manuel Resinas, Jan Mendling, and Antonio Ruiz Cortés. Automated team selection and compliance checking in business processes. In *ICSSP*, pages 42–51, 2015.

[8] Francesco Calimeri, Martin Gebser, Marco Maratea, and Francesco Ricca. Design and results of the fifth answer set programming competition. *Artificial Intelligence*, 231, 2016.

[9] Pedro M Castro and Inês Marques. Operating room scheduling with generalized disjunctive programming. *Computers & Operations Research*, 64:262–273, 2015.

[10] Thomas Eiter, Wolfgang Faber, Nicola Leone, Gerald Pfeifer, and Axel Polleres. Answer set planning under action costs. *J. Artif. Intell. Res. (JAIR)*, 19:25–71, 2003.

[11] Thomas Eiter, Giovambattista Ianni, Thomas Krennwallner, and Axel Polleres. Rules and Ontologies for the Semantic Web. In *Reasoning Web 2008*, volume 5224, pages 1–53. San Servolo Island, Venice, Italy, 2008.

[12] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. *Answer Set Solving in Practice*. Morgan & Claypool Publishers, 2012.

[13] Giray Havur, Cristina Cabanillas, Jan Mendling, and Axel Polleres. Automated Resource Allocation in Business Processes with Answer Set Programming. In *BPM Workshops (BPI)*, page In press, 2015.

[14] Niels Lohmann, Eric Verbeek, and Remco Dijkman. Petri Net Transformations for Business Processes - A Survey. *Transactions on Petri Nets and Other Models of Concurrency II*, 2:46–63, 2009.

[15] Michele Lombardi and Michela Milano. Optimal methods for resource allocation and scheduling: a cross-disciplinary survey. *Constraints*, 17:51–85, 2012.

[16] Ronny Mans, Nick C. Russell, Wil M. P. van der Aalst, Arnold J. Moleman, and Piet J. M. Bakker. Schedule-aware workflow management systems. *Trans. Petri Nets and Other Models of Concurrency*, 4:121–143, 2010.

[17] Wail Menesi, Mohamed Abdel-Monem, Tarek Hegazy, and Zinab Abuwarda. Multi-objective schedule optimization using constraint programming. In *ICSC15*, 2015.

[18] OMG. BPMN 2.0. Recommendation, OMG, 2011.

[19] Chun Ouyang, Moe Thandar Wynn, Colin Fidge, Arthur H.M. ter Hofstede, and Jan-Christian Kuhr. Modelling complex resource requirements in Business Process Management Systems. In *ACIS 2010*, 2010.

[20] Louchka Popova-Zeugmann. Time Petri Nets. In *Time and Petri Nets*, pages 139–140. Springer Berlin Heidelberg, 2013.

[21] Hajo A. Reijers, Irene T. P. Vanderfeesten, and Wil M. P. van der Aalst. The effectiveness of workflow management systems: A longitudinal study. *Int J. Information Management*, 36(1):126–141, 2016.

[22] Julia Rieck and Jürgen Zimmermann. Exact methods for resource leveling problems. In *Handbook on Project Management and Scheduling Vol. 1*. Springer, 2015.

[23] Atle Riise, Carlo Mannino, and Edmund K Burke. Modelling and solving generalised operational surgery scheduling problems. *Computers & Operations Research*, 66:1–11, 2016.

[24] Riivo Roose. Automated Resource Optimization in Business Processes. MSc. Thesis.

[25] Michael Rosemann and Jan vom Brocke. The six core elements of business process management. In *Handbook on Business Process Management 1*, pages 105–

122. Springer, 2015.

[26] Geary A Rummler and Alan J Ramias. A framework for defining and designing the structure of work. In *Handbook on Business Process Management 1*, pages 81–104. Springer, 2015.

[27] David S. Johnson and Michael R. Garey. Computers and Intractability: A Guide to the Theory of NP-Completeness. *WH Free. Co., San Fr*, 1979.

[28] Pinar Senkul and Ismail H. Toroslu. An Architecture for Workflow Scheduling Under Resource Allocation Constraints. *Inf. Syst.*, 30(5):399–422, July 2005.

[29] Thiago AO Silva, Mauricio C de Souza, Rodney R Saldanha, and Edmund K Burke. Surgical scheduling with simultaneous employment of specialised human resources. *European Journal of Operational Research*, 245(3):719–730, 2015.

[30] Ming-Fung Francis Siu, Ming Lu, and Simaan AbouRizk. Methodology for crew-job allocation optimization in project and workface scheduling. In *ASCE*, pages 652–659, 2015.

[31] Arno Sprecher and Andreas Drexl. Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm1. *European Journal of Operational Research*, 107(2):431 – 450, 1998.

[32] L. J. R. Stroppi, O. Chiotti, and P. D. Villarreal. A BPMN 2.0 Extension to Define the Resource Perspective of Business Process Models. In *CIbS'11*, 2011.

[33] Wil M. P. van der Aalst and Arthur H. M. ter Hofstede. YAWL: Yet Another Workflow Language. *Inf. Syst.*, 30(4):245–275, 2005.

[34] W.M.P. van der Aalst. Petri net based scheduling. *Operations-Research-Spektrum*, 18(4):219–229, 1996.

[35] Davy Van Nieuwenborgh, Martine De Cock, and Dirk Vermeir. Fuzzy answer set programming. In *Logics in Artificial Intelligence*, pages 359–372. Springer, 2006.

Published in the International Journal on Expert Systems with Applications (ESWA),
Volume 205, Number 117599, May 2022, Elsevier.

# Benchmarking Answer Set Programming Systems for Resource Allocation in Business Processes

Giray Havur[1,2], Cristina Cabanillas[3], and Axel Polleres[1,4]

[1]Vienna University of Economics and Business, Austria
{name.surname}@wu.ac.at
[2]Siemens AG Österreich, Technology, Vienna, Austria
{name.surname}@siemens.com
[3]SCORE Lab & I3US Institute, Universidad de Sevilla, Seville, Spain
{namesurname}@us.es
[4]Complexity Science Hub, Vienna, Austria

**Abstract.** Declarative logic programming formalisms are well-suited to model various optimization and configuration problems. In particular, Answer Set Programming (ASP) systems have gained popularity, for example, to deal with scheduling problems present in several domains. The main goal of this paper is to devise a benchmark for ASP systems to assess their performance when dealing with complex and realistic resource allocation with objective optimization. To this end, we provide (i) a declarative and compact encoding of the resource allocation problem in ASP (compliant with the ASP Core-2 standard), (ii) a configurable ASP systems benchmark named BRANCH that is equipped with resource allocation instance generators that produce problem instances of different sizes with adjustable parameters (e.g., in terms of process complexity, organizational and temporal constraints), and (iii) an evaluation of four state-of-the-art ASP systems using BRANCH. This solid application-oriented benchmark serves the ASP community with a tool that leads to potential optimizations and improvements in encodings and further drives the development of ASP solvers. On the other hand, resource allocation is an important problem that still lacks adequate automated tool support in the context of Business Process Management (BPM). The ASP problem encoding, ready-to-use ASP systems and problem instance generators benefit the BPM community to tackle the problem at scale and mitigate the lack of openly available problem instance data.

**Keywords:** Resource allocation, business process management, answer set programming, benchmark

## 1 Introduction

Business Process Management (BPM) is a discipline in operations management that aims at improving corporate performance by properly managing and optimizing a company's business processes. A business process is a collection of related events, activities and decisions that involve a number of human and non-human resources and that collectively lead to an outcome that is of value to an organization or its customers [1]. As an integral part of business processes, resources have to be considered throughout all the stages of the BPM life cycle, which iterates from process discovery and modeling to process execution and monitoring, targeting continuous process improvement.

At run time, a process execution engine creates business process instances and allocates specific resources to the tasks to be completed according to predefined criteria. Such criteria include, for example, constraints that comprise the characteristics of resources needed for each process task. Regarding human resources, these characteristics are usually reflected in organizational models that contain all the relevant data about the resources, such as their roles, skills and any other valuable information. For instance, Role-Based Access Control (RBAC) [2] is very often used for assigning resources to process tasks based on the organizational roles they are associated with. As a result, during process execution, at the due time for each activity only the required resource will be picked up from the set of candidates (i.e., among suitable resources according to the resource assignment constraints) and allocated to the task. Such a selection and scheduling of resource assignments is addressed in Resource Allocation in Business Processes (RABP).

An RABP problem comprises several components: a process model that describes the control flow of activities (e.g., precedence and concurrency relations), an organizational model that characterizes resources and their roles to enable activity executions [3], and a temporal model that designates activity duration estimations. Moreover, process-oriented organizations are also concerned with carrying out RABP under optimization objectives so that one or many process performance measures (e.g., execution time and cost) are optimized [1], which adds to the complexity of RABP. One way of approaching this problem is by encoding it using a declarative programming formalism. Answer Set Programming (ASP), a declarative logic programming dialect particularly suitable to model combinatorial search problems, appears to be a good candidate for this purpose as it has been used to solve various hard computational problems and proved to maintain a balance between expressiveness, ease of use and computational effectiveness [4]. ASP programs consist of clauses that look similar to Prolog rules [5] but the underlying computational mechanisms, which are based on the stable model semantics [6], are different [7]: rather than via resolution-based proofs, an ASP program is first *grounded* to a finite set of clauses where each rule with variables is instantiated as equivalent propositional rules without variables. Afterward, a *solver* searches for answer sets (i.e., solutions) of the ground program. This is typically done by relying on advanced conflict-driven, heuristic search procedures developed in the area of propositional satisfiability checking (SAT) [8]. Moreover, apart from rules and hard constraints, in the absence of complete information, *default behaviors* related to an allocated resource (e.g., assumed/default duration of a generic activity, potentially overridden by a known different duration for a specific resource/role) can be elegantly represented in ASP by so-called negation-as-failure [6, 7]. Due to its flexible modeling constructs, in prior research efforts [9, 10] we could demonstrate the suitability of ASP as a flexible tool to encode RABP with makespan[1] minimization, including relatively complex constraints. This allowed us to identify the RABP problem as a challenging task for state-of-the-art ASP systems (i.e., combinations of an ASP grounder and an ASP solver); however, so far, a comprehensive set of realistic benchmark instances is missing to stress-test the performance of ASP systems in practice on RABP.

[1] The makespan is the distance in time that elapses from the start to the end of a process execution.

The ASP community has collected benchmarks to test ASP systems by organizing regular competitions. These competitions have both been successful in driving research on more efficient grounders and solvers and optimizing and comparing problem encodings to evaluate the different performance of otherwise equivalent encodings in different ASP systems. Most closely related to our problem, earlier such ASP competitions [11, 12] have led to the development of two scheduling-related benchmark entries called disjunctive scheduling [12] and incremental scheduling [11]. Unlike these two problems that were addressed in two earlier ASP competitions, the RABP problem has a makespan optimization criteria and a more elaborate and expressive input/output setting that is tailored for real-world applicability, for which a benchmark to compare the performance of existing ASP solvers and grounders is still missing. In the case of RABP, this is also due to the lack of openly available real-world data describing various scenarios in which resource allocation is required (i.e., the lack of ready-to-use datasets), as companies typically consider information about their resources or business processes sensitive.

In this paper, we aim at narrowing this gap by extending former contributions on ASP-based RABP towards a common challenge benchmark for ASP solvers, driven by and parameterizable to mimic realistic scenarios from BPM. In particular, our contribution is three-fold:

- We define a formalization of the RABP problem and provide a *baseline problem encoding in ASP*.
- We develop a ready-to-use, configurable *benchmark* named BRANCH that generates RABP instances with respect to given parameters to adapt the instance sizes, solves the generated RABP instances using the configured ASP systems by the users, and reports on the performance of the ASP systems.
  We develop a ready-to-use, configurable *benchmark* named BRANCH that generates RABP instances with respect to given parameters to adapt the instance sizes, solves the generated RABP instances using the configured ASP systems by the users, and reports on the performance of the ASP systems.
- Lastly, by using BRANCH, we present a detailed *evaluation* that compares four ASP systems comprising combinations of state-of-the-art ASP grounders GRINGO [13] and I-DLV [14], and solvers CLASP [15] and WASP [16].

Interestingly, the results of our evaluation demonstrate that real-world instance sizes pose a considerable challenge on state-of-the-art ASP systems, which opens further opportunities for specific performance improvements in these systems. While this might be a valuable insight for the ASP research community, the BPM community will also benefit from a new, declarative encoding of the RABP problem, available systems to solve it, and a generator to produce instance datasets for testing purposes.

The remainder of this paper is structured as follows. Section 2 formally defines the RABP problem, providing the necessary background. Based on that, Section 3 presents our baseline problem encoding, preceded by a detailed background on ASP. Section 4 describes the functionalities of BRANCH. Section 5 evaluates the performance of different ASP systems using our baseline ASP encoding with BRANCH. Section 6 highlights the limitations of BRANCH. Finally, Section 7 concludes the paper, summarizing insights and giving pointers for future work.

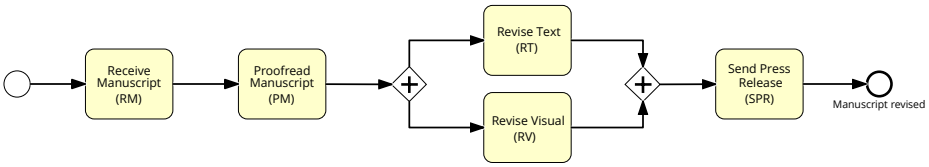



Fig. 1: BPMN model of the book publishing process.

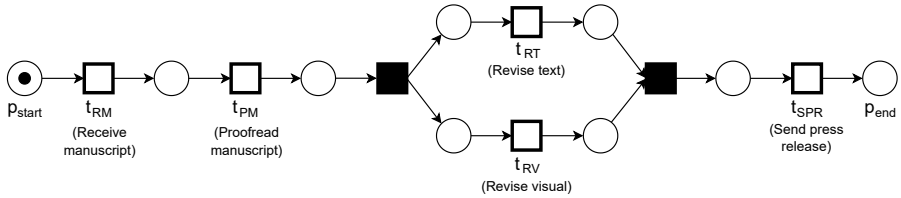



Fig. 2: Petri net model of the book publishing process.

## 2 Formalization of the Problem

In this section, we first describe the elements involved in RABP, such as business process models, organizational models, and the modeling of temporal aspects. Afterward, we formalize the RABP problem.

### 2.1 Business Process Models

A business process is a finite set of structured activities where a specific sequence of activities serves a particular business goal. The execution of each activity in the process generally requires one human resource and produces an output that is of benefit for a customer, such as a service or a product [17]. Process models are defined to represent the different aspects of a business process, especially the functional (process activities) and the behavioral (control flow or execution order) perspectives. As for formal descriptions of business processes, in practice, Business Process Model and Notation (BPMN) [18] diagrams are widely used to describe business processes due to their visual understandability.

***Book Publishing (BPub) Example 1.*** Figure Figure 1 depicts the model of a process for publishing a book from the point of view of a publishing house, represented as a BPMN diagram. In this process, when the publishing entity receives a new textbook manuscript from an author, it must be proofread and revised. The improved manuscript is then sent back to the author (press release) for double-checking.

High-level, visual process modeling notations like BPMN can be automatically mapped to Petri nets [19, 20], which have the advantage of an exact mathematical definition of their execution semantics and their well-developed theory for process analysis.

**Definition 1 (Petri Net)** *A* Petri net *is a 3-tuple* $PN = (P, T, F)$ *where:*
– $P = \{p_1, p_2, \ldots, p_n\}$ *is a set of* places*, represented graphically as circles;*

| | $RM$ | $PM$ | $RT$ | $RV$ | $SPR$ |
|---|---|---|---|---|---|
| $RM$ | | $\succ, \rightarrow$ | $\succ, \rightarrow$ | $\succ, \rightarrow$ | $\succ, \rightarrow$ |
| $PM$ | | | $\succ, \rightarrow$ | $\succ, \rightarrow$ | $\succ, \rightarrow$ |
| $RT$ | | | | $\succ, \|\|$ | $\succ, \rightarrow$ |
| $RV$ | | | $\succ, \|\|$ | | $\succ, \rightarrow$ |
| $SPR$ | | | | | |

Table 1: Behavioral relation matrix of the activities Figure 1.

- $T = \{t_1, t_2, \ldots, t_n\}$ *is a set of* transitions*, represented graphically as rectangles; and*
- $F \subseteq (P \times T) \bigcup (T \times P)$ *is a set of* arcs *(flow relations), represented as arrows and describing a bipartite graph.*

The *input places* and the *output places* of each transition $t \in T$ are defined as ${}^\bullet t = \{p \in P \mid (p,t) \in F\}$ and $t^\bullet = \{p \in P \mid (t,p) \in F\}$, respectively. Similarly, the *input transitions* and the *output transitions* of each place $p \in P$ are defined as ${}^\bullet p = \{t \in T \mid (t,p) \in F\}$ and $p^\bullet = \{t \in T \mid (p,t) \in F\}$, respectively. $\mu : P \rightarrow \mathbb{N}_0$ is a marking of $PN$ representing the initial distribution of tokens and $\mu(p)$ maps a place $p$ to its number of tokens. Pictorially, tokens are represented as black dots in places (e.g., $\mu(p_{start}) = 1$ in Figure 2). A transition $t$ is *enabled* in $\mu$, denoted by $(PN, \mu)[t\rangle$, when there is at least one token in each input place of $t$ where $p \in {}^\bullet t$. An enabled transition can therefore *fire*. The firing of a transition $t$ changes the current marking $\mu$ to $\mu'$ by removing one token from each $p \in {}^\bullet t$ and adding one token to each $p \in t^\bullet$, denoted by $(PN, \mu)[t\rangle(PN, \mu')$. A firing sequence of transitions $\sigma = t_1, t_2, \ldots, t_n$ changes the *initial marking* of the Petri net $\mu_0$ at each firing. If there exists a firing sequence $\sigma$ leading to $\mu'$, then $\mu'$ is *reachable* from $\mu_0$, denoted by $\mu' \in [PN, \mu_0\rangle$. A *Petri net system* is a pair $(PN, \mu_0)$.

***BPub Example 2.*** Figure 2 illustrates the Petri net representation of the book publishing process described above, corresponding to the BPMN diagram in Figure 1. Activities $A \subseteq T$ are transitions that are graphically represented by empty squares.

Among the *behavioral relations* that are described in [21], between two different transitions $t_x$ and $t_y$ of a Petri net system $(PN, \mu_0)$:

- $t_x \succ t_y$ reads as $t_x$ is in the *weak order relation* with $t_y$ if there exists a firing sequence $\sigma = t_1, t_2, \ldots, t_n$ such that $(PN, \mu_0)[\sigma\rangle$, $x \in \{1, \ldots, n-1\}$ and $x < y \leq n$;
- $t_x \rightarrow t_y$ reads as $t_x$ is in the *precedence relation* with (i.e., *precedes*) $t_y$
- $t_x \parallel t_y$ reads as $t_x$ is in the *concurrency relation* with $t_y$ if there is a reachable marking $\mu' \in [PN, \mu_0\rangle$ that enables both transitions concurrently.

***BPub Example 3.*** The behavioral relations between the activities in Figure 1 are derived from the equivalent Petri net in Figure 2. These relations are summarized in Table 1. For example, $PM \succ RT$, $PM \rightarrow RT$, and $RT \parallel RV$.

### 2.2 Organizational Model

Different types of organizational structures give rise to different organizational models [22]. In most cases, the employees of the organization (i.e., its human resources) have one or more organizational roles according to their skills and characteristics, which allow them to perform certain tasks and activities. One of the most widely known models for capturing such organizational structures based on roles is the Role-Based Access Control (RBAC) model [2]. It describes resources, roles, and *who* can execute *which* activity based on the roles.

**Definition 2 (RBAC Model)** *An* RBAC Model *is a 6-tuple* $O = (A, R, L, S_{AL}, S_{RL}, S_{LL})$*, where:*
- $A$ *is a set of* activities*,*
- $R$ *is a set of* resources*,*
- $L$ *is a set of* roles*,*
- $S_{AL} \subseteq 2^{(A \times L)}$ *is a set of activity-to-role assignments specifying which activity can be executed by which role(s) (i.e., the resources associated with these roles), commonly known as* resource assignment *in BPM [23],*
- $S_{RL} \subseteq 2^{(R \times L)}$ *is the corresponding set of resource-to-role assignment tuples identifying the roles per resource,*
- $S_{LL} \subseteq 2^{(L \times L)}$ *is a set of role-to-role assignments that form a hierarchical (sub-role) structure. The symbol* $\geq$ *indicates the ordering operator. If* $l_1 \geq l_2$*, then* $l_1$ *is referred to as the* senior *of* $l_2$*. Conversely,* $l_2$ *is the* junior *of* $l_1$ *with the intuitive meaning that each senior role may also execute all activities that the junior role is allowed to execute.*

***BPub Example 4.*** Let us assume that the book publishing process (cf. Figures 1 and 2) is executed within an organization composed of six resources $R = \{Amy, Glen, Drew, Emily, Oliver, Evan\}$ that are assigned to four distinct roles $L = \{Publisher, Copy\ Editor, Graphic\ Artist, Administrative\ Assistant\}$. Table 2 shows a possible RBAC model of such a publishing entity. For instance, within the *activity-to-role* relation, the first entry means that the activity *Receive Manuscript* can be executed by the members of the role *Publisher*; within the *resource-to-role* relation, the first entry means that the resource *Amy* has the role *Publisher*; and within the *role-to-role* relation, the first entry means that the resources with the role *Publisher* can execute all the activities of the resources with the role *Copy Editor* but not the other way around (specifically, *Publisher* $\geq$ *Copy Editor*).

### 2.3 Temporal Model

In a typical real-world BPM scenario, temporal constraints like durations of activities and the tentative deadline for the completion of process instances (i.e., makespan upper bound) are estimated by process managers.

**Definition 3 (Default Duration Function)** *The default duration function* $\Delta : A \rightarrow \mathbb{N}_0$ *defines the activity duration in a use-case specific time unit (TU) (e.g., in minutes, hours or days) for each activity* $a \in A$*.*

| *activity-to-role assignments* | |
|---|---|
| Receive Manuscript | Publisher |
| Proofread Manuscript | Copy Editor |
| Revise Text | Copy Editor |
| Revise Visual | Graphic Artist |
| Send Press Release | Administrative Assistant |
| *resource-to-role assignments* | |
| Amy | Publisher |
| Glen | Copy Editor |
| Drew | Copy Editor |
| Emily | Copy Editor |
| Oliver | Graphic Artist |
| Evan | Administrative Assistant |
| *role-to-role assignments* | |
| Publisher | Copy Editor |

Table 2: RBAC model of the publishing entity.

We may assume the temporal model is typically created by process managers.[2] However, to increase the robustness of resource allocation, *resource-activity*, and *role-activity* specific durations can be estimated from an event log using process mining techniques [24]. Process execution data from past process instances is typically stored in event logs or audit trails by BPM Systems (BPMS) and other information systems used in the organization [25]. An event log usually stores for each event related to an activity the type of event (e.g., start or end of an activity execution), resource-related properties (at least *who* executed an activity[3]), and temporal information (*when* the event took place). All the events related to a process instance constitute a *trace*. Each event and each trace in an event log are typically identified by a unique event id and trace id, respectively.

**Definition 4 (Resource-Activity Duration Partial Function)** *The resource-activity duration partial function* $\delta_r : (\mathcal{L} \times R \times A) \nrightarrow \mathbb{N}_0$ *estimates the mean duration for executing an activity* a *by a resource* r *from an event log* $\mathcal{L}$*.*

**Definition 5 (Role-Activity Duration Partial Function)** *The role-activity duration partial function* $\delta_l : (\mathcal{L} \times L \times A) \nrightarrow \mathbb{N}_0$ *estimates the mean duration for executing an activity* a *by a role* l *from an event log* $\mathcal{L}$ *where* $(r, l) \in S_{RL}$ *and* $(a, l) \in S_{AL}$*.*

***BPub Example 5.*** Table 3 shows an excerpt $\mathcal{L}'_{\text{BPub}}$ of the event log $\mathcal{L}_{\text{BPub}}$ from the execution of the book publishing process depicted in Figure 2. For example, the second row (with the event id e57) is logged at the start of the execution of the activity *Proofread Manuscript* by *Glen* on 2021-10-15 at 09:16. An example of role-activity duration estimation from an event log is as follows: given the partial event log $\mathcal{L}'_{\text{BPub}}$ in Table 3, $\pi(\mathcal{L}'_{\text{BPub}}, Copy\ Editor, Proofread\ Manuscript)$ is estimated by calculating

[2] Process managers design processes and implement improvements in them as needed.

[3] Within an organizational context, the resources that appear in the log are among those available according to the organizational model of the functional unit (e.g., an RBAC model).

| event_id | trace_id | event_type | activity | resource | time stamp |
|---|---|---|---|---|---|
| | | | ... | | |
| e56 | 7 | start | Receive Manuscript(RM) | Amy | 2021-10-15T**08:25** |
| e57 | 6 | start | Proofread Manuscript(PM) | Glen | 2021-10-15T**09:16** |
| e58 | 7 | end | Receive Manuscript(RM) | Amy | 2021-10-15T**09:21** |
| e59 | 7 | start | Proofread Manuscript(PM) | Drew | 2021-10-15T**09:35** |
| e60 | 7 | end | Proofread Manuscript(PM) | Drew | 2021-10-15T**12:45** |
| e61 | 7 | start | Revise Visual(RV) | Oliver | 2021-10-15T**12:55** |
| e62 | 7 | start | Revise Text(RT) | Drew | 2021-10-15T**13:07** |
| e63 | 8 | start | Receive Manuscript(RM) | Amy | 2021-10-15T**13:15** |
| e64 | 6 | end | Proofread Manuscript(PM) | Glen | 2021-10-15T**13:26** |
| e65 | 6 | start | Revise Text(RT) | Glen | 2021-10-15T**14:47** |
| | | | ... | | |

Table 3: The excerpt $\mathcal{L}'_{\text{BPub}}$ of the event log $\mathcal{L}_{\text{BPub}}$ from the execution of the book publishing process in Figure 2 executed by resources defined in the RBAC model in Table 2.

| *default activity durations* | |
|---|---|
| Receive Manuscript | 20 |
| Proofread Manuscript | 180 |
| Revise Text | 240 |
| Revise Visual | 240 |
| Send Press Release | 30 |
| *resource-activity durations* | |
| (Amy, Receive Manuscript) | 40 |
| (Drew, Proofread Manuscript) | 247 |
| (Glen, Proofread Manuscript) | 182 |
| (Drew, Revise Text) | 186 |
| (Glen, Revise Text) | 150 |
| (Oliver, Revise Visual) | 221 |
| (Evan, Send Press Release) | 55 |
| *role-activity durations* | |
| (Publisher, Receive Manuscript) | 45 |
| (Copy Editor, Proofread Manuscript) | 208 |
| (Copy Editor, Revise Text) | 171 |
| (Graphic Artist, Revise Visual) | 221 |
| (Administrative Asst., Send Press Release) | 55 |

Table 4: Default activity durations, resource-activity durations derived from the event log $\mathcal{L}_{\text{BPub}}$ and role-activity durations derived from the event log $\mathcal{L}_{\text{BPub}}$.

the mean of the two time intervals e57 to e64 and e59 to e60 (i.e., the copy editors *Glen* and *Drew*'s execution intervals of *Proofread Manuscript*). The first time interval is from 09:16 to 13:26 (250 min), and the second one is from 09:35 to 12:45 (190 min). Therefore, the mean of the two durations is estimated to be 220 min. Resource-activity duration estimations are calculated in a similar fashion. Associated with our running example, Table 4 shows the temporal model including the resource-activity and role-activity durations that are assumed to be extracted from the *complete* event log $\mathcal{L}_{\text{BPub}}$, and the default activity durations that are assumed to be estimated by the process manager while designing the process.

The selection of the duration to be considered in the allocation for a resource $r$'s execution of an activity $a$ is defined by the resource-activity duration preference function.

**Definition 6 (Resource-Activity Duration Preference Function)** *The resource-activity duration preference function* $\pi : (\boldsymbol{\mathcal{L}} \times R \times A) \rightarrow \mathbb{N}_0$ *handles the preference among the resource-activity duration* $\delta_r(\mathcal{L}, r, a)$*, role-activity duration* $\delta_l(\mathcal{L}, l, a)$*, and default duration of an activity* $\Delta(a)$*.*

$$\pi(\mathcal{L}, r, a) = \begin{cases} \delta_r(\mathcal{L}, r, a) & \text{if } \delta_r(\mathcal{L}, r, a) \text{ is defined,} \\ \delta_l(\mathcal{L}, l, a) & \text{if } \delta_r(\mathcal{L}, r, a) \text{ is not defined and} \\ & \quad (r, l) \in S_{RL}, \\ \Delta(a) & \text{otherwise.} \end{cases}$$

$\pi(\mathcal{L}, r, a)$ returns the resource-activity duration if $\delta_r(\mathcal{L}, r, a)$ has a value for the given parameters $\mathcal{L}$, $r$, and $a$. Otherwise, it returns the role-activity duration provided that $\delta_l(\mathcal{L}, l, a)$ has a value for the given parameters $\mathcal{L}$, $l$, and $a$ where there is an activity-to-role assignment $(r, l)$ in the RBAC model. When neither of the functions can be resolved, $\pi(\mathcal{L}, r, a)$ returns the default activity duration $\Delta(a)$, i.e., the estimated duration by the process manager while designing the process.

***BPub Example 6.*** $\pi(\mathcal{L}_{\text{BPub}}, Emily, PM)$ returns 208 time units since Emily has never executed *Proofread Manuscript* herself in the past (i.e., there is no resource-activity duration for (*Emily*, *Proofread Manuscript*) in Table 4), but there is a role-activity duration for (*Copy Editor*, *Proofread Manuscript*) where (*Emily*, *Copy Editor*) $\in S_{RL}$.

### 2.4 Resource Allocation in Business Processes (RABP)

RABP aims at finding a feasible allocation consisting of a set of quadruples $I \subseteq 2^{(R \times A \times U \times U)}$ such that $(r_i, a_i, s_i, c_i) \in I$ where each activity $a_i \in A$ is assigned a resource $r_i \in R$, a start time $s_i \in U$ and a completion time $c_i = s_i + \pi(\mathcal{L}, r, a)$. It is assumed that each activity, once started, is planned to be completed without interruptions in the schedule (i.e. activities are non-preemptive). The following constraints (c.1-c.4) hold for RABP:

(c.1) Only one resource is allocated to each activity.
$\forall a_i \in A : |\{(r_i, a_i, s_i, c_i) \in I\}| = 1$

(c.2) Each activity has only one start time.
$\forall i_1, i_2 \in I : a_{i_1} = a_{i_2} \Rightarrow s_{i_1} = s_{i_2}$

(c.3) The start time of any activity is greater than or equal to the completion time of its preceding activities.
$\forall i_1, i_2 \in I : a_{i_1} \rightarrow a_{i_2} \Rightarrow s_{i_2} \geq c_{i_1}$

(c.4) Same resource must not be allocated to any concurrent pair of activities that have overlapping execution periods.
$\forall i_1, i_2 \in I : (a_{i_1} || a_{i_2} \wedge s_{i_2} \leq s_{i_1} < c_{i_2}) \Rightarrow r_{i_1} \neq r_{i_2}$
$\forall i_1, i_2 \in I : (a_{i_1} || a_{i_2} \wedge s_{i_2} < c_{i_1} \leq c_{i_2}) \Rightarrow r_{i_1} \neq r_{i_2}$
$\forall i_1, i_2 \in I : (a_{i_1} || a_{i_2} \wedge s_{i_2} > s_{i_1} \wedge c_{i_2} < c_{i_1}) \Rightarrow r_{i_1} \neq r_{i_2}$

## 3 Baseline ASP Encoding of the Problem

Before we delve into the details of the encoding of RABP in ASP (Section 3.3), let us introduce fundamentals of ASP (Section 3.1) and some theoretical motivation of why RBAP is particularly suitable to be encoded and solved with ASP (Section 3.2).

### 3.1 Fundamentals of ASP

ASP [26] is a declarative (logic-programming-style) formalism for solving combinatorial search problems. An *ASP program* $\Pi$ is a finite set of logic programming rules of the form

$$a_0 :- a_1, \ldots, a_m, \textit{not } a_{m+1}, \ldots, \textit{not } a_n. \tag{1}$$

where $n \geq m \geq 0$ and each $a_i$ is a function-free first-order atom. The symbol ":-" is read as *if*: The left-hand side (e.g., $a_0$) is called the *head* of the rule and the right-hand side (e.g., $a_1, \ldots, a_m, \textit{not } a_{m+1}, \ldots, \textit{not } a_n$) is called the *body* of the rule. Semantically, *if* the *body* holds then the *head* is derived. When the *body* is empty, the symbol ":-" is dropped and the rule is called a *fact* (e.g., $a_0$.). "$not$" is called *negation as failure*: $not\ a_i$ is derived from failure to derive $a_i$. In rule (1), if $a_1, \ldots, a_m$ are true and none of *not* $a_{m+1}$, $\ldots$, *not* $a_n$ can be proven to be true then $a_0$ must be true. When the $head$ is empty in a rule, we call it a *constraint*. Constraints rule out models satisfying their body atoms, which eliminates unwanted solution candidates.

Whenever $a_i$ is a first-order predicate with variables within a rule $r$ of the form (1), this rule is considered a shortcut for its "grounding" $g(r)$ (i.e., the set of its instantiations obtained by replacing the variables with all possible constants occurring in $\Pi$). Likewise, we denote by $g(\Pi)$ the set of rules obtained from grounding all rules in $\Pi$.

As a syntactic extension, the ASP Core-2 standard [27] allows set-like *choice expressions* of the form

$$x \leq \{a_1, \ldots, a_m\} \leq y :- a_n, \ldots, a_k, \textit{not } a_{k+1}, \ldots, \textit{not } a_l.$$

that is, if the *body* holds then an arbitrary subset of $\{a_1, \ldots, a_m\}$ of minimum size of $x$ and maximum size of $y$ is derived.

***Example 3.1.*** The following ASP program describes a simplified resource allocation setting for exemplifying the concepts in ASP. In this resource allocation setting, the activities have no behavioral relation (i.e., precedence and concurrency relations) in contrast to RABP.

```
resource(r1). resource(r2). resource(r3).                    1
activity(a1;a2).                                             2

1<={allocation(R,A) : resource(R)}<=1                        3
:- activity(A).                                              4

:- allocation(R,A1), allocation(R,A2), A1!=A2.               5

:- allocation(r1,a2).                                        6
:- allocation(r2,a1).                                        7
```

Lines (1,2) are facts stating that `r1`, `r2` and `r3` are resources; and `a1` and `a2` are activities – "`activity(a1;a2).`" is a syntactic shortcut for "`activity(a1). activity(a2).`". Line (3) is a choice expression that generates the `allocation` predicates. This line can be read as follows: for each activity `a` $\in \{$`a1,a2`$\}$, derive minimum 1 and maximum 1 (i.e., only one) `allocation(r',a')` where `a'` is equal to `a`

and R is selected from the domain of the predicate resource (i.e., r' $\in$ {r1, r2, r3}). Lines (4–6) are constraints. Line (4) omits the answers in which a resource is allocated to more than one activity. Line (5) prohibits the resource r1 to be allocated to the activity a2, and line (6) prohibits the resource r2 to be allocated to the activity a1. The three different answers (i.e., solutions) of this program are as follows:

```
A 1: allocation(r1,a1) allocation(r2,a2)
A 2: allocation(r3,a1) allocation(r2,a2)
A 3: allocation(r1,a1) allocation(r3,a2)
```

*Aggregates* [28] are arithmetic operations over a set of elements and they occur in aggregate atoms in rule bodies that have the form $\#aggr\{w_1 : a_1, \ldots, w_n : a_n\} \oplus v$, where $w_i$ is the weight assigned to $a_i$; the operation $aggr \in$ {"count" ,"sum","min", "max"}; the comparison operator $\oplus \in$ {"$<$","$\leq$","=","$\neq$", "$>$","$\geq$"}; and $v$ is a bound value. For instance, the aggregate $\#\texttt{sum}\{w_1 : a_1, \ldots, w_n : a_n\} \oplus v$ is interpreted as $true$ if $\sum_{i=1}^{n} w_i \oplus v$ holds, and $false$ otherwise. In addition, an aggregate on the right-hand side of the assignment operator "=" may also be used for assigning the result obtained from the aggregate's evaluated value to a variable.

***Example 3.2.*** Following the Example 3.1, we add the following lines in our program for taking into account the execution times required for each resource to execute an activity.

```
execTime(r1,a1,10). execTime(r1,a2,10).                          7
execTime(r2,a1,20). execTime(r2,a2,20).                          8
execTime(r3,a1,30). execTime(r3,a2,30).                          9

totalTime(T) :-
    T=#sum{E, A : allocation(R,A), execTime(R,A,E)}.             10
```

Lines (7–9) represent the execution times of activities by specific resources. For example, execTime(r1,a1,10) means that the resource r1 executes the activity a1 in 10 min. Line (10) sums the total execution time of all activities with respect to the allocated resources. The program described in lines (1–10) returns the following three answers:

```
A 1: allocation(r1,a1) allocation(r2,a2) totalTime(30)
A 2: allocation(r1,a1) allocation(r3,a2) totalTime(40)
A 3: allocation(r3,a1) allocation(r2,a2) totalTime(50)
```

*Weak constraints* [29] allow us to formalize optimization problems in an easy and natural way. In such problems we use weak constraints to indicate preferences between possible answer sets:

$$:\sim a_1, \ldots, a_m.\ [w, t_1, \ldots, t_n]$$

where $t_1, \ldots, t_n$ are terms (e.g., atoms), and $w$ is a weight. The sum of weights $w$ over all occurrences of weighted atoms that are satisfied by a stable model are therefore minimized.

***Example 3.3.*** We add a weak constraint to our example as follows:

```
:~ totalTime(T). [T, T]                                          11
```

Line (11) asks the solver to minimize the variable `T` of `totalTime` in the answer. The program described in lines (1–11) returns the following optimum answer:

```
OPT: allocation(r1,a1) allocation(r2,a2) totalTime(30)
```

Therefore, the optimum answer with the minimum value of the `totalTime` variable `T` is returned as an answer (cf. Example 3.2).

Sets of rules are evaluated in ASP under the *stable-model semantics* [6], which allows several models (so-called *answer sets*) . The ASP systems typically first compute a $g(\Pi)$ via a *grounder*, and then use a Davis–Putnam–Logemann–Loveland- (DPLL)-like branch-and-bound algorithm to find answer sets for this ground program via a *solver*. There are various *grounders* and *solvers* for solving problems encoded in ASP [30]. They vary in terms of the meta-heuristics and extensions incorporated in their implementations. We refer to [26] and references therein for details.

### 3.2 Theoretical motivation: Why to solve RABP with ASP?

In order to motivate why the RABP problem as introduced in this paper is amenable to be solved with ASP, let us argue by comparing RABP to related problems from the literature.

A couple of scheduling-related benchmarks have been presented and executed in the earlier ASP competitions [11, 12]. The first entry is called disjunctive scheduling [12] and assumes generalized activity precedence relations (i.e., any activity can be preceded by any other activities without further limitations, such as those imposed in RABP due to the need of having well-formed business process structures), uniform activity durations (i.e., all activities have the same execution time), and an overall deadline to be met without additional resource-related constraints. The other related entry is called incremental scheduling [11] and addresses a problem with one renewable resource[4] set, fixed activity durations (i.e., each activity has only one possible execution time) and deadlines imposed on activities. Unlike the problems involved in these two ASP competition entries, the RABP problem has different numbers of resource sets (i.e., roles), resource- and resource-set-specific time requirements and – as the overall optimization criterion – minimization of the total makespan.

Allocation of resources with starting times to the activities (i.e., RABP) is a far-from-trivial task with strong implications on the quality of the final allocation, and it is incredibly challenging from a computational perspective [31]. As for complexity, RABP with makespan optimization is NP-Hard: the Job Shop Scheduling Problem (JSSP), known to be NP-Complete for its variants with two or more machines [32], is polynomial-time reducible to a RABP problem. A JSSP instance can simply be translated into a RABP problem instance where (i) the tasks of different jobs that require the same machine would be represented as activities in concurrency relation, (ii) the order of tasks in jobs maps to precedence relations of activities, (iii) machines map to roles with only one resource, and (iv) task durations for specific machines are represented as default activity durations. Since we provide an encoding of RABP in ASP with weak

[4] Renewable resources are available with a constant amount in each time period, e.g., human resources.

```
% selection of a makespan value
makespanDomain(U) :- upperBound(U).                                          12
makespanDomain(M1) :- makespanDomain(M), M1=M-1, M1>=0.                      13
1<={makespan(M) : makespanDomain(M)}<=1.                                     14

% time domain generation from the makespan
time(0).                                                                     15
time(T1) :- time(T), T1=T+1, T1<=U, makespan(U).                             16

% senior roles can execute activities of junior roles
alAC(A,L1) :- llAC(L1,L2), alAC(A,L2).                                       17

% resource-activity duration interval preference
defRAD(R,A,D) :- raDuration(R,A,D), rlAC(R,L), alAC(A,L).                    18
defRAD(R,A,D) :- not raDuration(R,A,_), laDuration(L,A,D), rlAC(R,L),        19
     alAC(A,L).
defRAD(R,A,D) :- not raDuration(R,A,_), not laDuration(L,A,_),               20
     defActDuration(A,D), rlAC(R,L), alAC(A,L).

% (c.1)
1<={allocation(R,A,S,C): time(S), time(C), defRAD(R,A,D), C=S+D}<=1 :-       21
     activity(A).

% (c.2)
:- allocation(_,A,S1,_), allocation(_,A,S2,_), S1<S2.                        22

% (c.3)
:- prec(A1,A2), allocation(_,A1,_,C1), allocation(_,A2,S2,_), C1>S2.         23

% (c.4)
:- conc(A1,A2), allocation(R,A1,S1,_), allocation(R,A2,S2,C2), S2<=S1,       24
     C2>S1, A1<A2.
:- conc(A1,A2), allocation(R,A1,_,C1), allocation(R,A2,S2,C2), S2<C1,        25
     C2>=C1, A1<A2.
:- conc(A1,A2), allocation(R,A1,S1,C1), allocation(R,A2,S2,C2), S2>S1,       26
     C2<C1, A1<A2.

% minimization of the makespan
:~ makespan(M). [M,M]                                                        27
```

Fig. 3: ASP encoding for the RABP problem.

constraints, which captures the class of $\Delta_2^P$ that is typically used for computing optimal solutions among such NP-complete problems with a given upper bound $u$ to be minimized [33], we also show that RABP is in $\Delta_2^P$.

Moreover, ASP provides a simple and elegant representation to define and handle the preferences/priorities of resource-bound activity durations and also default activity durations in absence of complete information. For these reasons, therefore, the RABP problem seems to be an ideal candidate for benchmarking ASP systems supporting weak constraints.

### 3.3 ASP Encoding of RABP

Our encoding is shown in Figure 3. It encapsulates the formalization in Section 2.4 in a straightforward, declarative manner (i.e., no encoding "tricks" that optimize for a specific ASP grounder or solver have been applied).

**Input Format.** The input is divided into three groups of predicates as follows.

| *Process Model* | |
|---|---|
| activity | (rm; pm; rt; rv; spr) |
| prec | (rm,pm; rm,rt; rm,rv; rm,spr; pm,rt; pm,rv; pm,spr; rt,spr; rv,spr) |
| conc | (rt,rv; rv,rt) |

| *Organizational Model* | |
|---|---|
| alAC | (rm,publ; pm,copyEd; rt,copyEd; rv,graphAr; spr,adAsst) |
| rlAC | (amy,publ; glen,copyEd; drew,copyEd; emily,copyEd; oliver,graphAr; evan,adAsst) |
| llAC | (publ,copyEd) |

| *Temporal Model* | |
|---|---|
| defActDuration | (rm,20; pm,180; rt,240; rv,240; spr,30) |
| raDuration | (amy,rm,40; drew,pm,247; glen,pm,182; drew,rt,186; glen,rt,150; oliver,rv,221; evan,spr,55) |
| laDuration | (publ,rm,45; copyEd,pm,208; copyEd,rt,171; graphAr,rv,221; adAsst,spr,55) |
| upperBound | (350) |

Table 5: ASP encoding of the book publishing example described in Section 2.

*Behavioral relations of activities in a business process $\mathcal{P}$ as described in Section 2.1:*

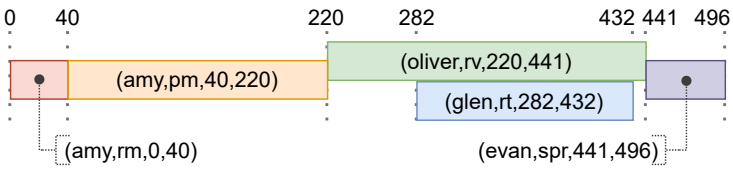
- `activity`($a$): $a$ is an activity in $\mathcal{P}$;
- `prec`($a_1$, $a_2$): the activity $a_1$ precedes the activity $a_2$;
- `conc`($a_1$, $a_2$): the activity $a_1$ is concurrency with the activity $a_2$.

*An RBAC organizational model $O = (A, R, L, S_{AL}, S_{RL}, S_{LL})$ as described in Section 2.2:*

- `alAC`($a$, $l$): the resources with role $l$ can execute activity $a$ (i.e., $(a, l) \in S_{AL}$);
- `rlAC`($r$, $l$): resource $r$ has role $l$ (i.e., $(r, l) \in S_{RL}$);
- `llAC`($l_1$, $l_2$): the resources with role $l_1$ can execute the same activities as the resources with role $l_2$ (i.e., $(l_1, l_2) \in S_{LL}$).

*A temporal model as described in Section 2.3:*

- `defActDuration`($a$, $d$): the default duration of activity $a$ is estimated as $d$ (i.e., $\Delta(a) = d$);




**OPT:** allocation(amy,rm,0,40; amy,pm,40,220; glen,rt,282,432; oliver,rv,220,441; evan,spr,441,496)

Fig. 4: Optimal solution of the book publishing example in Section 2.

- `raDuration`($r$, $a$, $d$): the duration of activity $a$ is estimated as $d$ when it is executed by resource $r$ from $\mathcal{L}_\mathcal{P}$ (i.e., $\delta_r(\mathcal{L}_\mathcal{P}, r, a) = d$);
- `laDuration`($l$, $a$, $d$): the duration of activity $a$ is estimated as $d$ when it is executed by a resource that has role $l$ from $\mathcal{L}_\mathcal{P}$ (i.e., $\delta_l(\mathcal{L}_\mathcal{P}, l, a) = d$);
- `upperBound`($u$): makespan is bounded at $u$ time units.

The code reads as follows. Rules (12–14) select a makespan for the allocation. Rules (15, 16) generate the time domain from the selected makespan. Rule (17) propagates the permissions of activity executions of a junior role to a senior role (i.e., role-to-role RBAC relations). Rules (18–20) implement the resource-activity duration preference handling mechanism described in Definition 6. Rules (21), (22), (23), and (24–26) correspond to the constraints (c.1), (c.2), (c.3), and (c.4) described in Section 2.4, respectively. Finally, Rule (27) minimizes the makespan.

**Output Format.** For the allocation output we use the following predicate as described in Section 2.4:

- `allocation`($r$, $a$, $s$, $c$): a resource $r$ is allocated to activity $a$ at the start time $s$ until the completion time $c$ (i.e., $(r, a, s, c) \in I$).

***BPub Example 7.*** The book publishing example described in Section 2 (the process model from Figure 1 and Table 1, the organizational model from Table 2, and the temporal model from Table 4) is encoded in ASP as depicted in Table 5. Its optimal solution computed by an ASP system using the RABP encoding in Figure 3 is shown in Figure 4. The solution can be read as the following: *Amy* is allocated to *Receive Manuscript* at time 0 and it is planned in the resultant allocation that *Amy* takes 40 min to execute this activity. Then, *Amy* is allocated to *Proofread Manuscript* with a starting time of 40 min, and she takes 180 min to execute the activity, and so on. .

## 4 Implementation of the ASP Systems Benchmark for RABP

The lack of datasets for benchmarking ASP systems for RABP, unlike other related scheduling domains [34, 35, 36], led us to design and implement a ready-to-use benchmark BRANCH that generates RABP problem instances and solves them via user-configured ASP Systems [37]. BRANCH is implemented in a user-friendly fashion with

| *Petri net generator* | | *RBAC generator* | | *Temporal knowledge generator* | |
|---|---|---|---|---|---|
| Num. of activities | $n_A$ | Num. of resources | $n_R$ | Upper bound | $u$ |
| Degree of concurrency | $\psi_{conc}$ | Num. of roles | $n_L$ | Num. of resource-activity durations | $n_{d_{RA}}$ |
| | | | | Num. of role-activity durations | $n_{d_{LA}}$ |

Table 6: Parameters for generating one RABP problem instance.

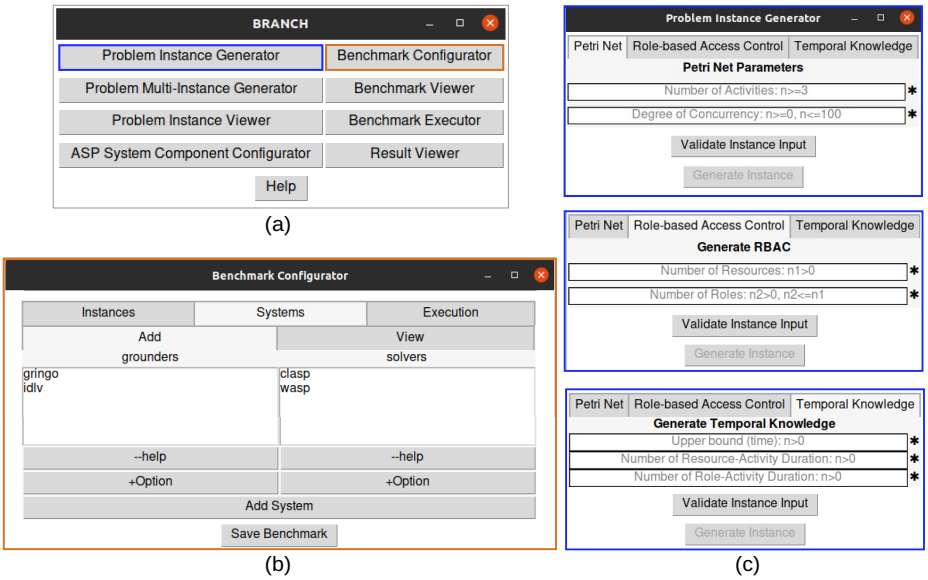
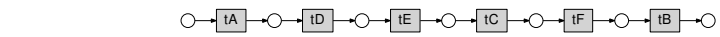



Fig. 5: Screenshots of the benchmark.

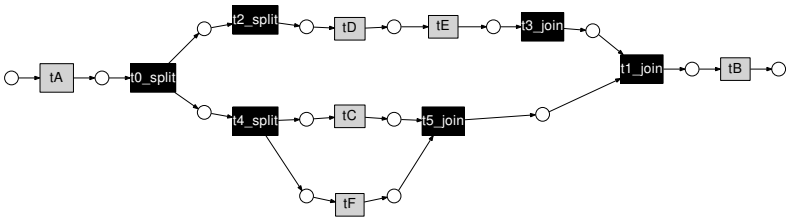
a simple User Interface (UI)[5]. Figure 5(a) shows the main menu of BRANCH. The functionalities provided within BRANCH are further described below.

**Problem (Multi-)Instance Generator:** This component involves a process model, an organizational model and a temporal knowledge generator, as depicted in Figure 5(c). An overview of required parameters for generating one RABP problem instance is summarized in Table 6. The inputs of the process model generator are the number of activities and the degree of concurrency of the generated process model. BRANCH uses the *stochastic Petri net* plug-in of the process mining tool ProM [38] for generating Petri nets. This plug-in performs a series of random structured insertions of control-flow constructs resulting in a random Petri net that is sound, free-choice and block-structured [39]. For example, Figure 6 shows three generated Petri nets with different degree of concurrency. After a Petri net is generated, the behavioral relations of its activities are derived for RABP. An organizational model is then generated using the RBAC model [2] by entering a number of resources and a number of roles that are necessary to create the model. The RBAC model consists of a resource set, a role set, activity-to-role

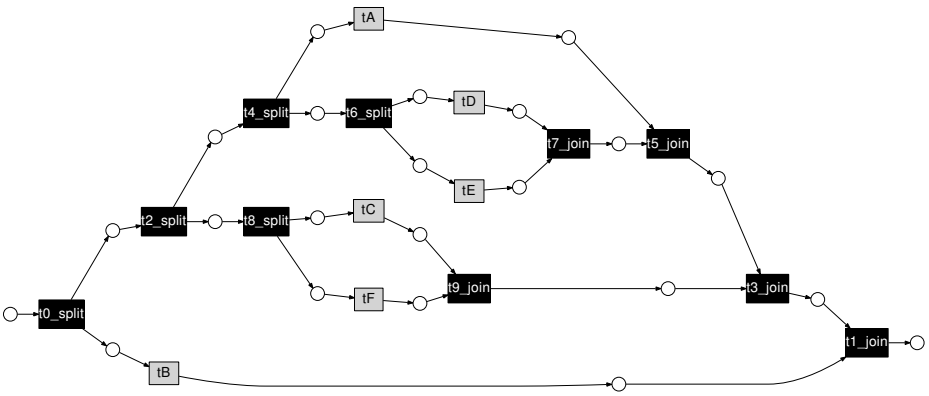
[5] The UI is implemented via the Python appJar module.




(a) $\psi_{conc} = 0$




(b) $\psi_{conc} = 50$




(c) $\psi_{conc} = 100$

Fig. 6: Three generated Petri nets with 6 activities.

assignment tuples specifying which activity can be executed by the resources associated with which role(s), and tuples of resource-to-role assignments identifying the roles of a resource. Finally, the temporal knowledge is generated given an upper bound for the execution of the process, resource-activity specific durations, and role-activity specific durations (apart from default activity durations). Users can generate multiple problem instances at once via the *multi-instance* generator. Each of the previously described numeric inputs are parameterized by three values (lower limit, mode, and upper limit). Therefore, their respective triangular random variables are sampled to generate multiple RABP instances.

**Problem Instance (resp. Benchmark) Viewer:** Generated problem instances (resp. benchmarks) are listed in a table including the parameters used for their generation. The user can also remove the instances.

**ASP System Component Configurator:** The components of the ASP systems are added by naming the component, its type and the executable path. We include the state-of-the-art ASP *grounders* GRINGO [13] and I-DLV [14]; and ASP *solvers* CLASP [15]

and WASP [16] in BRANCH for the sake of convenience of the benchmark user. These grounders and solvers are selected mainly due to being among the top-performing tools in the latest ASP Competitions [11, 30].

**Benchmark Configurator:** To set up a new benchmark instance the user can select problem instances to include as well as add new ASP systems to be used by selecting and customizing (e.g., adding command line options) grounder+solver configurations with the UI shown in Figure 5(b). A custom problem encoding for RABP can also be selected if desired and the execution details of the benchmark (e.g., time and memory limitations) can be further constrained.

**Benchmark Executor:** The user can select one benchmark instance to execute. The grounding and solving instances are dynamically listed in the monitored interface.

**Result Viewer:** The time and memory usage statistics of ASP grounders and solvers can be exported in *csv* format. We also implemented box plots and cactus plots for an easier interpretation of the benchmark results.

BRANCH follows the applicable basic principles of experimental design by providing (i) *randomization* via the multi-instance generator, and (ii) *replication* via the benchmark configurator and benchmark executor.

The benchmark software, a video that screencasts the tool usage and a separate tutorial document are available at `https://urban.ai.wu.ac.at/~havur/eswa2022/`.

## 5 BRANCH in Use

As aforementioned, BRANCH is configurable in the sense that any ASP grounder and ASP solver can be added and their optional parameters are textually given for purposes like activating their meta-heuristics and ensuring the compatibility between grounders and solvers, when possible. We demonstrate the functionalities of BRANCH via configuring four different ASP systems using out-of-the-box grounders and solvers: GRINGO+CLASP (i.e., CLINGO [40]), GRINGO+WASP, I-DLV+CLASP, and I-DLV+WASP (i.e., DLV 2.0 [41]). We use the most up-to-date versions of these grounders and solvers: GRINGO 5.5.0, I-DLV 1.1.6, CLASP 3.3.6, WASP 2.0. These tools are executed on their default meta-heuristic options (i.e., no extra parameter is given) in this benchmark except for the combination of the grounder GRINGO and the solver WASP. While configuring the ASP system GRINGO+WASP, we set the output format of GRINGO to *smodels* [42] by simply adding "`-o smodels`" parameter for GRINGO using the `+Option` button in Figure 5(b) because GRINGO's default output format *aspif* [43] is not compatible with WASP.

**Platform:** The benchmark has been run on an Ubuntu Linux server (64 bit), equipped with a 16 Core 2.40 GHz Intel Xeon Processor and 128 GB RAM. Time and memory for each run were limited to 2 h CPU clock time and 20 GB, respectively.

**Problem Instances:** We generated 70 problem instances using BRANCH's instance generator. The details of these instances are provided in Table 7. In the table, *id* is the unique identifier of each instance, $n_A$ is the number of activities to which resources are going to be allocated, $\psi_{conc}$ is the degree of concurrency of the generated Petri net, $n_R$

Table 7: Properties of problem instances.

| id | $n_A$ | $\delta_{conc}$ | $n_R$ | $n_L$ | $SAT$ | $u$ | id | $n_A$ | $\delta_{conc}$ | $n_R$ | $n_L$ | $SAT$ | $u$ | id | $n_A$ | $\delta_{conc}$ | $n_R$ | $n_L$ | $SAT$ | $u$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 8 | 50 | 2 | 1 | $Yes$ | 90 | **24** | 16 | 30 | 16 | 4 | $Yes$ | 90 | **47** | 32 | 90 | 16 | 8 | $Yes$ | 330 |
| **2** | 8 | 75 | 4 | 1 | $Yes$ | 70 | **25** | 16 | 30 | 16 | 4 | $Yes$ | 85 | **48** | 32 | 80 | 16 | 8 | $Yes$ | 260 |
| **3** | 8 | 100 | 4 | 1 | $Yes$ | 105 | **26** | 16 | 90 | 16 | 4 | $Yes$ | 140 | **49** | 32 | 90 | 16 | 8 | $Yes$ | 310 |
| **4** | 8 | 100 | 8 | 2 | $Yes$ | 70 | **27** | 16 | 30 | 37 | 8 | $Yes$ | 65 | **50** | 32 | 90 | 16 | 8 | $Yes$ | 360 |
| **5** | 16 | 90 | 2 | 1 | $Yes$ | 200 | **28** | 16 | 50 | 35 | 8 | $Yes$ | 75 | **51** | 32 | 50 | 17 | 4 | $Yes$ | 145 |
| **6** | 16 | 50 | 4 | 1 | $Yes$ | 75 | **29** | 16 | 75 | 35 | 8 | $Yes$ | 130 | **52** | 32 | 50 | 17 | 4 | $Yes$ | 145 |
| **7** | 16 | 90 | 8 | 4 | $Yes$ | 175 | **30** | 16 | 90 | 34 | 8 | $Yes$ | 190 | **53** | 32 | 50 | 32 | 16 | $Yes$ | 160 |
| **8** | 16 | 50 | 4 | 1 | $Yes$ | 120 | **31** | 32 | 90 | 2 | 1 | $Yes$ | 330 | **54** | 32 | 80 | 16 | 4 | $Yes$ | 330 |
| **9** | 16 | 90 | 4 | 1 | $Yes$ | 185 | **32** | 32 | 95 | 2 | 1 | $Yes$ | 360 | **55** | 32 | 50 | 33 | 8 | $Yes$ | 380 |
| **10** | 16 | 75 | 4 | 1 | $Yes$ | 145 | **33** | 32 | 30 | 4 | 1 | $Yes$ | 100 | **56** | 32 | 90 | 32 | 16 | $Yes$ | 345 |
| **11** | 16 | 90 | 4 | 1 | $Yes$ | 210 | **34** | 32 | 60 | 4 | 1 | $Yes$ | 210 | **57** | 32 | 90 | 17 | 4 | $Yes$ | 350 |
| **12** | 16 | 90 | 9 | 2 | $Yes$ | 205 | **35** | 32 | 75 | 8 | 4 | $No$ | 210 | **58** | 32 | 50 | 32 | 16 | $Yes$ | 235 |
| **13** | 16 | 50 | 8 | 2 | $Yes$ | 130 | **36** | 32 | 60 | 4 | 1 | $Yes$ | 275 | **59** | 32 | 50 | 16 | 4 | $Yes$ | 360 |
| **14** | 16 | 75 | 8 | 2 | $Yes$ | 75 | **37** | 32 | 30 | 4 | 1 | $Yes$ | 270 | **60** | 32 | 80 | 32 | 16 | $Yes$ | 290 |
| **15** | 16 | 40 | 16 | 8 | $Yes$ | 85 | **38** | 32 | 85 | 4 | 1 | $Yes$ | 270 | **61** | 32 | 60 | 32 | 16 | $Yes$ | 520 |
| **16** | 16 | 40 | 8 | 2 | $Yes$ | 75 | **39** | 32 | 85 | 4 | 1 | $Yes$ | 300 | **62** | 32 | 50 | 33 | 8 | $Yes$ | 150 |
| **17** | 16 | 90 | 16 | 8 | $Yes$ | 190 | **40** | 32 | 90 | 4 | 1 | $Yes$ | 145 | **63** | 32 | 80 | 34 | 8 | $Yes$ | 317 |
| **18** | 16 | 90 | 8 | 2 | $Yes$ | 210 | **41** | 32 | 90 | 8 | 4 | $Yes$ | 355 | **64** | 32 | 90 | 33 | 8 | $Yes$ | 340 |
| **19** | 16 | 75 | 8 | 2 | $Yes$ | 115 | **42** | 32 | 90 | 8 | 4 | $Yes$ | 360 | **65** | 64 | 90 | 8 | 4 | $No$ | 305 |
| **20** | 16 | 75 | 16 | 8 | $Yes$ | 120 | **43** | 32 | 30 | 8 | 2 | $Yes$ | 200 | **66** | 64 | 90 | 8 | 4 | $No$ | 310 |
| **21** | 16 | 90 | 16 | 8 | $Yes$ | 165 | **44** | 32 | 75 | 16 | 8 | $Yes$ | 280 | **67** | 64 | 60 | 16 | 8 | $Yes$ | 295 |
| **22** | 16 | 90 | 9 | 2 | $Yes$ | 200 | **45** | 32 | 90 | 8 | 2 | $Yes$ | 315 | **68** | 64 | 90 | 16 | 8 | $Yes$ | 710 |
| **23** | 16 | 30 | 17 | 4 | $Yes$ | 75 | **46** | 32 | 50 | 16 | 8 | $Yes$ | 170 | **69** | 64 | 60 | 64 | 32 | $Yes$ | 320 |
| | | | | | | | | | | | | | | **70** | 64 | 90 | 64 | 32 | $Yes$ | 620 |

Table 8: Grounder statistics.

| | GRINGO | | | I-DLV | | | | GRINGO | | | I-DLV | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **id** | *time* | *mem* | $\|g(\Pi)\|$ | *time* | *mem* | $\|g(\Pi)\|$ | **id** | *time* | *mem* | $\|g(\Pi)\|$ | *time* | *mem* | $\|g(\Pi)\|$ |
| **1** | 3 | 7 | 29 | 3 | 74 | 20 | **36** | 1689 | 22 | 16323 | 960 | 4403 | 4526 |
| **2** | 7 | 8 | 74 | 4 | 86 | 25 | **37** | 1572 | 23 | 15544 | 1049 | 4272 | 4347 |
| **3** | 19 | 8 | 163 | 11 | 174 | 56 | **38** | 1459 | 18 | 15766 | 855 | 4284 | 4363 |
| **4** | 7 | 7 | 85 | 3 | 129 | 28 | **39** | 1860 | 20 | 19515 | 1107 | 5158 | 5422 |
| **5** | 68 | 9 | 527 | 76 | 598 | 330 | **40** | 460 | 15 | 4465 | 265 | 1300 | 1218 |
| **6** | 30 | 9 | 316 | 15 | 193 | 91 | **41** | 672 | 15 | 6743 | 1176 | 8736 | 3818 |
| **7** | 30 | 9 | 287 | 52 | 737 | 188 | **42** | 445 | 15 | 6939 | 925 | 8969 | 3927 |
| **8** | 77 | 9 | 752 | 46 | 439 | 239 | **43** | 513 | 18 | 8369 | 491 | 3552 | 2376 |
| **9** | 192 | 10 | 1858 | 116 | 1024 | 570 | **44** | 233 | 16 | 4192 | 26 | 150 | 292 |
| **10** | 112 | 10 | 1129 | 66 | 643 | 350 | **45** | 1245 | 24 | 21315 | 1307 | 8285 | 6056 |
| **11** | 222 | 11 | 2301 | 137 | 1226 | 700 | **46** | 94 | 11 | 1519 | 10 | 92 | 115 |
| **12** | 291 | 13 | 2977 | 252 | 2003 | 809 | **47** | 356 | 15 | 5810 | 37 | 174 | 394 |
| **13** | 95 | 11 | 972 | 87 | 821 | 311 | **48** | 205 | 15 | 3593 | 23 | 142 | 252 |
| **14** | 29 | 9 | 297 | 5 | 179 | 46 | **49** | 291 | 14 | 5129 | 31 | 164 | 350 |
| **15** | 8 | 8 | 97 | 1 | 30 | 14 | **50** | 463 | 18 | 6912 | 44 | 187 | 465 |
| **16** | 27 | 9 | 307 | 5 | 180 | 46 | **51** | 290 | 13 | 5130 | 20 | 96 | 182 |
| **17** | 43 | 10 | 456 | 6 | 54 | 67 | **52** | 309 | 16 | 5530 | 21 | 97 | 191 |
| **18** | 259 | 11 | 2449 | 227 | 1870 | 751 | **53** | 78 | 11 | 1334 | 8 | 141 | 111 |
| **19** | 75 | 9 | 684 | 65 | 595 | 222 | **54** | 1283 | 25 | 23428 | 2266 | 14761 | 6651 |
| **20** | 21 | 8 | 184 | 3 | 39 | 27 | **55** | 2011 | 28 | 32896 | 120 | 373 | 1073 |
| **21** | 31 | 9 | 344 | 4 | 49 | 49 | **56** | 374 | 17 | 6341 | 35 | 293 | 447 |
| **22** | 242 | 11 | 2817 | 151 | 1869 | 735 | **57** | 1724 | 21 | 29670 | 1508 | 18263 | 8335 |
| **23** | 37 | 8 | 345 | 3 | 32 | 25 | **58** | 154 | 12 | 2900 | 16 | 207 | 217 |
| **24** | 45 | 9 | 438 | 3 | 34 | 32 | **59** | 1643 | 27 | 27747 | 1431 | 18011 | 8234 |
| **25** | 40 | 8 | 398 | 3 | 33 | 29 | **60** | 253 | 16 | 4495 | 25 | 251 | 325 |
| **26** | 87 | 10 | 1025 | 17 | 567 | 157 | **61** | 807 | 19 | 14386 | 83 | 456 | 993 |
| **27** | 29 | 9 | 308 | 2 | 42 | 26 | **62** | 282 | 13 | 5158 | 20 | 154 | 201 |
| **28** | 32 | 9 | 351 | 3 | 45 | 30 | **63** | 1461 | 26 | 24932 | 85 | 325 | 800 |
| **29** | 109 | 10 | 1161 | 8 | 74 | 84 | **64** | 1624 | 26 | 26446 | 102 | 335 | 871 |
| **30** | 244 | 11 | 2214 | 17 | 96 | 148 | **65** | 1120 | 26 | 19233 | 93 | 198 | 726 |
| **31** | 591 | 14 | 6122 | 489 | 3151 | 3320 | **66** | 1268 | 59 | 20685 | 114 | 204 | 768 |
| **32** | 814 | 15 | 6860 | 776 | 3688 | 3744 | **67** | 1094 | 26 | 18091 | 86 | 308 | 678 |
| **33** | 246 | 12 | 2278 | 159 | 717 | 638 | **68** | – | 37 | 93938 | 512 | 737 | 3847 |
| **34** | 1092 | 15 | 9483 | 669 | 2598 | 2582 | **69** | 1242 | 22 | 21228 | 78 | 981 | 925 |
| **35** | 232 | 12 | 2199 | 21 | 72 | 161 | **70** | 5118 | 45 | 80788 | 279 | 1907 | 3130 |

Table 9: Solver statistics.

| | | CLASP(GRINGO) | | | WASP(GRINGO) | | | CLASP(I-DLV) | | | WASP(I-DLV) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **id** | $u$ | *time* | *mem* | $c_{max}$ | *time* | *mem* | $c_{max}$ | *time* | *mem* | $c_{max}$ | *time* | *mem* | $c_{max}$ |
| **1** | 90 | 315 | 279 | ***69*** | 533 | 680 | ***69*** | 345 | 169 | ***69*** | 465 | 415 | ***69*** |
| **2** | 70 | 73 | 581 | ***54*** | 107 | 1487 | ***54*** | 12 | 200 | ***54*** | 37 | 485 | ***54*** |
| **3** | 105 | 720 | 1426 | ***79*** | 1535 | 3588 | ***79*** | 498 | 439 | ***79*** | 415 | 855 | ***79*** |
| **4** | 70 | 97 | 733 | ***55*** | 371 | 1875 | ***55*** | 21 | 257 | ***55*** | 50 | 702 | ***55*** |
| **5** | 200 | – | 5964 | 200 | – | 14000 | 192 | 3923 | 1865 | ***149*** | – | 3639 | 179 |
| **6** | 75 | – | 2947 | 68 | – | 7220 | 68 | – | 543 | 68 | 3037 | 1109 | ***68*** |
| **7** | 175 | 2305 | 3360 | ***133*** | 4788 | 7821 | ***133*** | 1217 | 1449 | ***133*** | 2971 | 3459 | ***133*** |
| **8** | 120 | – | 7912 | 98 | – | 18600 | 101 | – | 1302 | 92 | – | 2650 | 92 |
| **9** | 185 | 5302 | – | | 192 | 19007 | | – | 3196 | | – | 6245 | 176 |
| **10** | 145 | 7146 | 12026 | | 106 | – | | 5279 | 1941 | ***108*** | 5014 | 3908 | ***108*** |
| **11** | 210 | 6646 | – | | 207 | – | | – | 3876 | | – | 7567 | |
| **12** | 205 | – | 11788 | | 315 | – | | – | 5156 | 192 | – | 10920 | |
| **13** | 130 | – | 10487 | 134 | 96 | – | | 2025 | 2042 | ***104*** | 2084 | 4368 | ***104*** |
| **14** | 75 | 1785 | 3190 | ***59*** | 2278 | 7411 | ***59*** | 84 | 467 | ***59*** | 133 | 983 | ***59*** |
| **15** | 85 | 115 | 951 | ***64*** | 148 | 2317 | ***64*** | 3 | 116 | ***64*** | 17 | 234 | ***64*** |
| **16** | 75 | 935 | 2985 | ***56*** | 432 | 6962 | ***56*** | 34 | 456 | ***56*** | 56 | 1003 | ***56*** |
| **17** | 190 | 5134 | 5547 | ***144*** | 7157 | 12500 | 160 | 68 | 432 | ***144*** | 279 | 418 | ***144*** |
| **18** | 210 | – | 10295 | | 267 | – | | – | 4825 | 206 | – | 10081 | |
| **19** | 115 | – | 8690 | 109 | – | 18098 | 100 | 377 | 1482 | ***86*** | 929 | 3162 | ***86*** |
| **20** | 120 | 479 | 2067 | ***92*** | 889 | 4860 | ***92*** | 13 | 205 | ***92*** | 33 | 279 | ***92*** |
| **21** | 165 | 1739 | 4026 | ***124*** | 4562 | 9450 | ***124*** | 37 | 344 | ***124*** | 111 | 373 | ***124*** |
| **22** | 200 | – | 11283 | | 253 | – | | – | 4753 | 160 | – | 9817 | 195 |
| **23** | 75 | 1766 | 3395 | ***64*** | 1730 | 8062 | ***64*** | 10 | 202 | ***64*** | 25 | 283 | ***64*** |
| **24** | 90 | 4032 | 4319 | ***64*** | 2545 | 10498 | ***64*** | 25 | 234 | ***64*** | 57 | 299 | ***64*** |
| **25** | 85 | 1785 | 3793 | ***67*** | 2171 | 9018 | ***67*** | 14 | 217 | ***67*** | 30 | 289 | ***67*** |
| **26** | 140 | – | 11780 | | 83 | – | | 599 | 1571 | ***106*** | 1609 | 2888 | ***106*** |
| **27** | 65 | 1060 | 2995 | ***48*** | 388 | 7016 | ***48*** | 9 | 242 | ***48*** | 16 | 433 | ***48*** |
| **28** | 75 | 1433 | 3664 | ***56*** | 1013 | 8255 | ***56*** | 17 | 260 | ***56*** | 83 | 449 | ***56*** |
| **29** | 130 | – | 12586 | 122 | 101 | – | | 91 | 609 | ***99*** | 183 | 803 | ***99*** |
| **30** | 190 | – | 9359 | | 260 | – | | 650 | 1026 | ***143*** | 933 | 1082 | ***143*** |
| **31** | 330 | 621 | – | | 1010 | – | | – | 11872 | | 284 | – | |
| **32** | 360 | 670 | – | | 1168 | – | | – | 13344 | | 297 | – | |
| **33** | 100 | – | 10537 | | 258 | – | | – | 2467 | | – | 5751 | |
| **34** | 210 | 635 | – | | 1006 | – | | – | 9483 | | 287 | – | |
| **35** | 210 | 5856 | – | | 234 | – | | 271 | 919 | | 411 | 553 | |
| **36** | 275 | 605 | – | | 928 | – | | – | 16237 | | 312 | – | |
| **37** | 270 | 553 | – | | 938 | – | | – | 15679 | 145 | 272 | – | |
| **38** | 270 | 582 | – | | 869 | – | | – | 15746 | | 243 | – | |
| **39** | 300 | 635 | – | | 1070 | – | | 997 | – | | 221 | – | |
| **40** | 145 | 618 | – | | 555 | – | | – | 4612 | | – | 10675 | |
| **41** | 355 | 634 | – | | 1017 | – | | 796 | – | | 161 | – | |
| **42** | 360 | 406 | – | | 795 | – | | 565 | – | | 145 | – | |
| **43** | 200 | 366 | – | | 628 | – | | – | 10440 | 76 | 170 | – | |
| **44** | 280 | – | 17891 | | 396 | – | | 545 | 1770 | ***210*** | 1119 | 1258 | ***210*** |
| **45** | 315 | 366 | – | | 660 | – | | 803 | – | | 124 | – | |
| **46** | 170 | – | 17411 | | 87 | – | | 57 | 769 | ***125*** | 170 | 701 | ***125*** |
| **47** | 330 | 414 | – | | 793 | – | | 2194 | 2289 | ***250*** | 3650 | 2063 | ***250*** |
| **48** | 260 | – | 14796 | | 307 | – | | 499 | 1559 | ***197*** | 1034 | 1079 | ***197*** |
| **49** | 310 | 386 | – | | 573 | – | | 1241 | 2069 | ***233*** | 2896 | 1824 | ***233*** |
| **50** | 360 | 391 | – | | 786 | – | | 1998 | 2818 | ***271*** | 4946 | 2654 | ***271*** |
| **51** | 145 | 356 | – | | 450 | – | | 197 | 1142 | ***110*** | 288 | 954 | ***110*** |
| **52** | 145 | 347 | – | | 506 | – | | 469 | 1194 | ***118*** | 666 | 1017 | ***118*** |
| **53** | 160 | – | 15506 | 150 | 94 | – | | 58 | 846 | ***122*** | 204 | 1076 | ***122*** |
| **54** | 330 | 414 | – | | 757 | – | | 790 | – | | 132 | – | |
| **55** | 380 | 397 | – | | 755 | – | | 2286 | 6601 | ***163*** | 2644 | 4370 | ***163*** |
| **56** | 345 | 415 | – | | 723 | – | | 1975 | 2953 | ***258*** | 4291 | 2695 | ***258*** |
| **57** | 350 | 369 | – | | 672 | – | | 456 | – | | 47 | – | |
| **58** | 235 | 6300 | – | | 183 | – | | 102 | 1522 | ***132*** | 553 | 1580 | ***132*** |
| **59** | 360 | 384 | – | | 770 | – | | 465 | – | | 48 | – | |
| **60** | 290 | 352 | – | | 448 | – | | 655 | 2120 | ***215*** | 1622 | 1962 | ***215*** |
| **61** | 520 | 399 | – | | 812 | – | | 677 | 5929 | ***123*** | 2685 | 3665 | ***123*** |
| **62** | 150 | 357 | – | | 479 | – | | 266 | 1437 | ***126*** | 366 | 1631 | ***126*** |
| **63** | 317 | 394 | – | | 769 | – | | 5339 | 4903 | ***238*** | 5881 | 4326 | ***238*** |
| **64** | 340 | 412 | – | | 769 | – | | – | 5619 | 313 | – | 4394 | 274 |
| **65** | 305 | 414 | – | | 783 | – | | 1640 | 3865 | | 2002 | 1933 | |
| **66** | 310 | 396 | – | | 713 | – | | 2072 | 4057 | | 2397 | 2091 | |
| **67** | 295 | 416 | – | | 769 | – | | 2424 | 3908 | ***221*** | 3945 | 2545 | ***221*** |
| **68** | 710 | | | | | | | 481 | – | | – | 7885 | |
| **69** | 320 | 395 | – | | 760 | – | | 5794 | 6307 | ***238*** | – | 7863 | 313 |
| **70** | 620 | 378 | – | | 838 | – | | 7152 | – | | – | 16188 | |

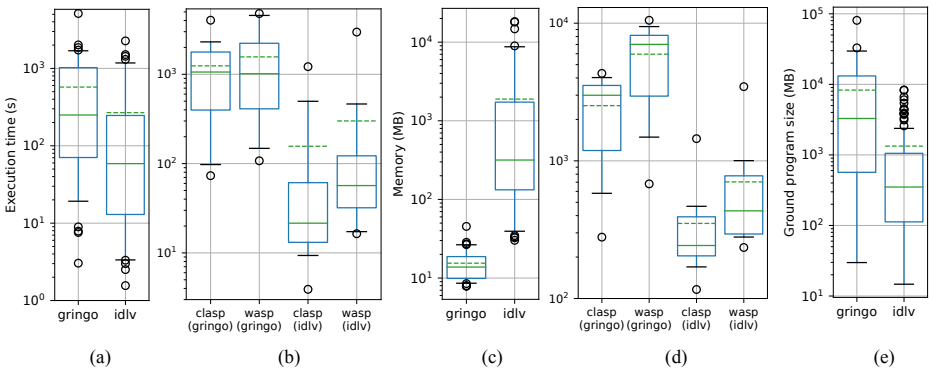
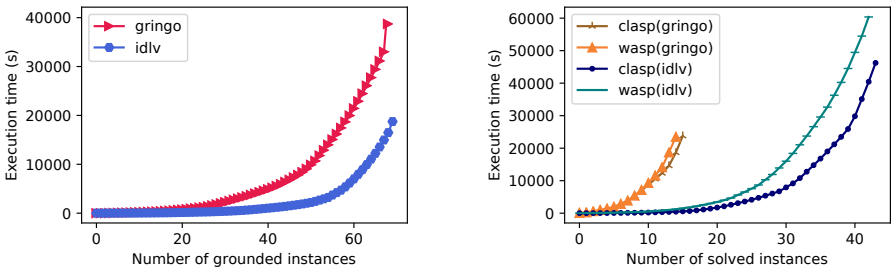



Fig. 7: Box plots regarding the performance statistics of grounders and solvers.




(a) Grounding times

(b) Solving times

Fig. 8: Sorted cactus plots.

is the number of resources, and $n_L$ is the number of roles. The $SAT$ column indicates if there exists a feasible solution for the problem instance in the given upper bound $u$.

We ran the benchmark and summarized the results in Tables 8 and 9. Table 8 shows the performance statistics of two grounders while grounding the problem instances. In the table, *time* is the CPU time in seconds, *mem* is the memory used in MB, and $g(\Pi)$ is the size of the ground program in MB. Table 9 presents the performance statistics of only the ASP solvers. CLASP(GRINGO) indicates the performance of the solver CLASP using the ground program from the grounder GRINGO. $c_{max}$ is the makespan (i.e., maximum of the activity completion times) computed by the system, which is minimized by the weak constraint. Within this column, bold and italic numbers are the confirmed optima cases; and other values are the instances for which a solution is found but the solution is not proven to be the optimal one (e.g., when the time or the memory limit is reached). For both tables "–" within the $time$ column means "out-of-time" and "–" within the $mem$ column means "out-of-memory".

The box plots in Figure 7 visually summarize Tables 8 and 9. We logarithmically scaled the $y$-axis of the plots for the sake of readability. 90% of the samples are in between the upper and lower whiskers. The solid green line represents the *median*, and the dashed green line represents the *mean* of the sample. Figure 7(a) and Figure 7(b)

compare the CPU execution times of grounders and solvers. Figure 7(c) and Figure 7(d) compare the memory usage of grounders and solvers. Figure 7(e) reports on the ground program sizes of the instances. By looking at Figure 7(a) and Figure 7(c), we find out that I-DLV grounds the problem instances much faster than GRINGO, although it has a larger memory footprint. The program rewriting (i.e., intelligent projections) feature of I-DLV is a vital optimization for this problem as the ground programs that are generated by I-DLV are much smaller than those generated by GRINGO (cf. Table 8 and Figure 7(e)).

The cactus plots in Figure 8 separately compare the grounder and solver performances. Less steep curves (cf. Figure 8(b)) and smaller memory footprint (cf. Figure 7(d)) of CLASP(GRINGO) and CLASP(I-DLV) – in comparison to those of WASP(GRINGO) and WASP(I-DLV) – illustrate that the ASP solver CLASP performs better than WASP. To address the makespan minimization, both solvers contain several solving methods: model-guided methods aim to produce models with descending costs, and core-guided methods identify and relax unsatisfiable cores until a model is found [15, 16]. In the default setting, CLASP uses its branch-and-bound algorithm [44], while WASP utilizes a core-guided algorithm [45]. Results suggest that both solvers behave non-optimally, mainly because the given ASP encoding for the RABP problem is rather tailored towards a declarative and readable set of rules than for a performance gain of particular underlying solver techniques. Therefore, we see room for improvement not only in solver techniques but also in bespoke problem encoding optimizations, as they have been successfully applied to other ASP benchmark problems in the past [11].

Overall, I-DLV+CLASP completes 41 instances within the given time and memory constraints whereas I-DLV+WASP, GRINGO+CLASP, and GRINGO+WASP complete 40, 16, and 15 instances, respectively. This result shows that I-DLV+CLASP is the most performant ASP system of our baseline benchmark for RABP.

## 6 Limitations

The RABP problem is characterized by the essential ideas behind designing and managing business processes and organizational models [46, 47, 1]. Problems similar to RABP have been widely acknowledged in research areas other than Logic Programming (LP), such as Constraint Programming (CP), Machine Learning (ML) and Operations Research (OR) [48, 31]. The survey conducted in [49] presents that automatic resource allocation in business processes has a wide variety of implementations encompassing a subset of capabilities for:

- **allocation mode** (one resource to one activity, one resource to many activities, many resources to one activity, and many resources to many activities),
- **optimization goal** (finding best-fitting resource, makespan minimization, cost minimization, balancing workload among the resources, etc.),
- **resource taxonomies and attributes** (previous performance, workload, role, expertise, etc.), and
- **type of evaluation** (simulation experiments, experiments with real-world data, case study, etc.).

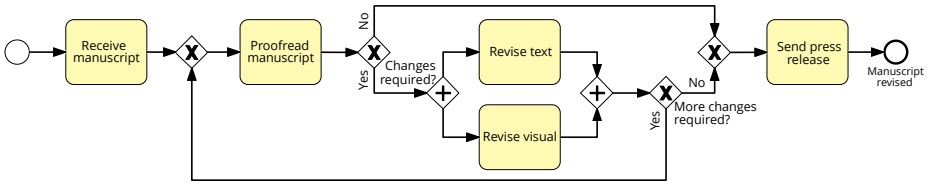



Fig. 9: BPMN model of the book publishing process with decision nodes.

This survey also shows the statistics of the studies that focus on these capabilities. To find the right balance in the complexity of the problem definition and to keep the focus on the ASP system benchmark BRANCH, we implemented the most studied capabilities: allocation of *one resource to one activity*, *finding best-fitting resource*, *makespan minimization*, *role-based resource taxonomy* (i.e., RBAC model), and *performance-based resource attributes* (resource- and role-based resource-activity durations) as described in Section 2. In our previous work, we also addressed resource allocation with extended capabilities, namely, a Petri net firing semantics-based resource allocation [9] that *simulates* the behavior of the Petri net, and an extended encoding that includes further requirements [10], such as allocation of *many resources to one activity*, allocation of *non-human resources*, *expertise* levels of resources, and *cost minimization*. Such capabilities could also be included in BRANCH with a reasonable degree of effort, provided that ASP encodings are already available.

Figure 9 depicts the BPMN model of the book publishing process including decision points. The main difference between the running example model in Figure 1 and this model is that, in this process model, after proofreading the manuscript, *if changes are required* the modifications suggested must be applied on text and figures. This review-and-improvement *loop* is *repeated* until there are no more changes to apply. In real-world scenarios, such decision points are crucial for describing the processes. However, this kind of uncertainty in the execution time of processes causally divides the search space for RABP (i.e., the complete set of activities to execute RABP cannot be known before execution). To address this issue, first, a decision-node-free process fragment (i.e., partial-run) needs to be generated under a number of assumptions on these decision nodes. For example, given the book publishing process including the two decision points in Figure 9, when it is assumed that *changes required* at the first decision node (the branch labeled with *yes*) is taken, and *no more changes are required* at the second decision node (the branch labeled with *no*) is taken, the decision-node-free process fragment in Figure 1 is generated. In case an assumption does not hold at run time, a new decision-node-free process fragment (only from the decision point onward) is generated, and a new RABP instance is triggered (i.e., resource reallocation). Our previous work investigated the most effective and robust way of generating decision-node-free process fragments from process models with decision nodes for RABP [50]. As RABP cannot be performed when there is a decision node in the process, we omit the decision nodes in the formalization of RABP in Section 2, and also in the problem instance generator in Section 4.

## 7 Conclusions

We have formalized the RABP problem and provided a new benchmark for ASP systems. The results of the illustrative benchmark run show that RABP is a challenging problem for the ASP systems that are among the most performant in the previous ASP Competitions [11, 30]. This application-oriented benchmark would be beneficial to the ASP community by helping assess advances in the formalism (e.g., the ease of encoding – i.e., the compactness, readability, modularity and maintainability of the problem encoding) and the computational performance of the solvers; and further, encourage the BPM community to integrate and test RABP in the process execution environments.

Future work will involve optimizing the ASP encoding of RABP to improve its computational efficiency to operate in large-scale real-world scenarios. It is also on our agenda to extend BRANCH towards more flexible grounding and solving paradigms such as *multi-shot ASP solving* [51] (e.g., to support the use of an incremental encoding of RABP) and formalisms other than ASP. Moreover, devising an interface to existing BPMSs from BRANCH might prove useful in building and deploying tailored solutions for resource allocation needs in BPM.

## Acknowledgements

This work has been partially supported by: (i) European Regional Development Fund (ERDF)-A way of making Europe, (ii) CONFLEX project (grant number RTI2018-100763-J-I00) funded by Ministerio de Ciencia e Innovación – Agencia Estatal de Investigación (MCIN/AEI/10.13039/501100011033), and (iii) MEMENTO project (grant number US-1381595) funded by Programa Operativo FEDER 2014-2020 and Junta de Andalucía (Consejería de Economía, Conocimiento, Empresas y Universidad). Axel Polleres' work is supported by TEAMING.AI project (grant number 957402) funded by the European Union's Horizon 2020 research and innovation program.

## References

[1] Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A. Reijers. *Fundamentals of Business Process Management*. Springer, 2 edition, 2018.

[2] Alessandro Colantonio, Roberto Di Pietro, Alberto Ocello, and Nino Vincenzo Verde. A formal framework to elicit roles with business meaning in RBAC systems. In Barbara Carminati and James Joshi, editors, *Proceedings of the 14th ACM Symposium on Access Control Models and Technologies, SACMAT 2009*, pages 85–94. ACM, 2009.

[3] Nick Russell, Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and David Edmond. Workflow resource patterns: Identification, representation and tool support. In Oscar Pastor and João Falcão e Cunha, editors, *Proceedings of the 17th International Conference on Advanced Information Systems Engineering, CAiSE 2005*, volume 3520 of *Lecture Notes in Computer Science*, pages 216–232. Springer, 2005.

[4] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

[5] Alain Colmerauer and Philippe Roussel. The birth of Prolog. In John A. N. Lee and Jean E. Sammet, editors, *Preprints of the 2nd ACM SIGPLAN Conference on History of Programming Languages Conference (HOPL-II)*, pages 37–52. ACM, 1993.

[6] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert A. Kowalski and Kenneth A. Bowen, editors, *Proceedings of the 5th International Conference and Symposium on Logic Programming*, pages 1070–1080. MIT Press, 1988.

[7] Vladimir Lifschitz. What is answer set programming? In Dieter Fox and Carla P. Gomes, editors, *Proceedings of the 23rd AAAI Conference on Artificial Intelligence, AAAI 2008*, pages 1594–1597. AAAI Press, 2008.

[8] Christian Drescher, Martin Gebser, Benjamin Kaufmann, and Torsten Schaub. Heuristics in conflict resolution. *The Computing Research Repository (CoRR), arXiv*.

[9] Giray Havur, Cristina Cabanillas, Jan Mendling, and Axel Polleres. Automated resource allocation in business processes with answer set programming. In Manfred Reichert and Hajo A. Reijers, editors, *Revised Papers of the 13th International Business Process Management Workshops, BPM 2015*.

[10] Giray Havur, Cristina Cabanillas, Jan Mendling, and Axel Polleres. Resource allocation with dependencies in business process management systems. In Marcello La Rosa, Peter Loos, and Oscar Pastor, editors, *Proceedings of the Business Process Management Forum, BPM Forum 2016*, volume 260 of *Lecture Notes in Business Information Processing*, pages 3–19. Springer, 2016.

[11] Martin Gebser, Marco Maratea, and Francesco Ricca. The seventh answer set programming competition: Design and results. *Theory and Practice of Logic Programming*, 20(2):176–204, 2020.

[12] Marc Denecker, Joost Vennekens, Stephen Bond, Martin Gebser, and Miroslaw Truszczynski. The second answer set programming competition. In Esra Erdem, Fangzhen Lin, and Torsten Schaub, editors, *Proceedings of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR 2009*.

[13] Martin Gebser, Roland Kaminski, Arne König, and Torsten Schaub. Advances in gringo series 3. In James P. Delgrande and Wolfgang Faber, editors, *Proceedings of the 11th International Conference on Logic Programming and Nonmonotonic Reasoning LPNMR 2011*, volume 6645 of *Lecture Notes in Computer Science*, pages 345–351. Springer, 2011.

[14] Francesco Calimeri, Davide Fuscà, Simona Perri, and Jessica Zangari. I-DLV: the new intelligent grounder of DLV. *Intelligenza Artificiale*, 11(1):5–20, 2017.

[15] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Javier Romero, and Torsten Schaub. Progress in clasp series 3. In Francesco Calimeri, Giovambattista Ianni, and Miroslaw Truszczynski, editors, *Proceedings of the 13th International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR 2015*, volume 9345 of *Lecture Notes in Computer Science*, pages 368–383. Springer, 2015.

[16] Mario Alviano, Carmine Dodaro, Nicola Leone, and Francesco Ricca. Advances in WASP. In Francesco Calimeri, Giovambattista Ianni, and Miroslaw Truszczynski, editors, *Proceedings of the 13th International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR 2015*, volume 9345 of *Lecture Notes in Computer Science*, pages 40–54. Springer, 2015.

[17] Andrea Burattin. *Introduction to Business Processes, BPM, and BPM Systems*, pages 11–21. Springer International Publishing, Cham, 2015.

[18] OMG. Business Process Model and Notation (BPMN), 2014. Specification.

[19] James Lyle Peterson. *Petri net theory and the modeling of systems*. Englewood Cliffs, N.J., Prentice-hall, 1981.

[20] Remco M. Dijkman, Marlon Dumas, and Chun Ouyang. Semantics and analysis of business process models in BPMN. *Information and Software Technology*, 50(12):1281–1294, 2008.

[21] Matthias Weidlich, Jan Mendling, and Mathias Weske. Efficient consistency measurement based on behavioral profiles of process models. *IEEE Transactions on Software Engineering*, 37(3):410–429, 2011.

[22] Bryan Horling and Victor R. Lesser. A survey of multi-agent organizational paradigms. *Knowledge Engineering Review*, 19(4):281–316, 2004.

[23] Cristina Cabanillas, Manuel Resinas, Adela del-Río-Ortega, and Antonio Ruiz Cortés. Specification and automated design-time analysis of the business process human resource perspective. *Information Systems*, 52:55–82, 2015.

[24] Wil M. P. van der Aalst. *Process Mining - Data Science in Action*. Springer, 2 edition, 2016.

[25] Wil M. P. van der Aalst and Boudewijn F. van Dongen. Discovering petri nets from event logs. *Transactions on Petri Nets and Other Models of Concurrency*, 7:372–422, 2013.

[26] Gerhard Brewka, Thomas Eiter, and Mirosław Truszczyński. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2011.

[27] Francesco Calimeri, Wolfgang Faber, Martin Gebser, Giovambattista Ianni, Roland Kaminski, Thomas Krennwallner, Nicola Leone, Marco Maratea, Francesco Ricca, and Torsten Schaub. ASP-Core-2 input language format. *Theory and Practice of Logic Programming*, 20(2):294–309, 2020.

[28] Martin Gebser, Roland Kaminski, Max Ostrowski, Torsten Schaub, and Sven Thiele. On the input language of ASP grounder gringo. In Esra Erdem, Fangzhen Lin, and Torsten Schaub, editors, *Proceedings of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR 2009*.

[29] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic*, 7(3):499–562, 2006.

[30] Martin Gebser, Marco Maratea, and Francesco Ricca. The sixth answer set programming competition. *Journal of Artificial Intelligence Research*, 60:41–95, 2017.

[31] Michele Lombardi and Michela Milano. Optimal methods for resource allocation and scheduling: a cross-disciplinary survey. *Constraints*, 17(1):51–85, 2012.

[32] Michael R. Garey, David S. Johnson, and Ravi Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1(2):117–129, 1976.

[33] Francesco Buccafurri, Nicola Leone, and Pasquale Rullo. Adding weak constraints to disjunctive datalog. In Moreno Falaschi, Marisa Navarro, and Alberto Policriti, editors, *Proceedings of the Joint Conference on Declarative Programming, APPIA-GULP-PRODE'97*, pages 557–568, 1997.

[34] Sonda Elloumi, Taïcir Loukil, and Philippe Fortemps. Reactive heuristics for disrupted multi-mode resource-constrained project scheduling problem. *Expert Systems with Applications*, 167:114132, 2021.

[35] Hongbo Li and Xuebing Dong. Multi-mode resource leveling in projects with mode-dependent generalized precedence relations. *Expert Systems with Applications*, 97:193–204, 2018.

[36] Dimitris C. Paraskevopoulos, Christos D. Tarantilis, and George Ioannou. Solving project scheduling problems with resource constraints via an event list-based evolutionary algorithm. *Expert Systems with Applications*, 39(4):3983–3994, 2012.

[37] Giray Havur, Cristina Cabanillas, and Axel Polleres. BRANCH: An ASP systems benchmark for resource allocation in business processes. In Wil M. P. van der Aalst, Remco M. Dijkman, Akhil Kumar, Francesco Leotta, Fabrizio Maria Maggi, Jan Mendling, Brian T. Pentland, Arik Senderovich, Marcos Sepúlveda, Estefanía Serral Asensio, and Mathias Weske, editors, *Proceedings of the Best Dissertation Award, Doctoral Consortium, and Demonstration & Resources Track at BPM 2021*, volume 2973 of *CEUR Workshop Proceedings*, pages 176–180. CEUR-WS.org, 2021.

[38] Andreas Rogge-Solti. Stochastic Petri net plug-in of the process mining framework ProM.

[39] W. M. P. van der Aalst. *Structural characterizations of sound workflow nets*, volume 9623.

[40] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub.

[41] Mario Alviano, Francesco Calimeri, Carmine Dodaro, Davide Fuscà, Nicola Leone, Simona Perri, Francesco Ricca, Pierfrancesco Veltri, and Jessica Zangari. The ASP system DLV2. In Marcello Balduccini and Tomi Janhunen, editors, *Proceedings of the 14th International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR 2017*, volume 10377 of *Lecture Notes in Computer Science*, pages 215–221. Springer, 2017.

[42] Tommi Syrjänen. *Lparse 1.0 user's manual*. 2000.

[43] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Max Ostrowski, Torsten Schaub, and Philipp Wanko. Theory solving made easy with clingo 5. In Manuel Carro, Andy King, Neda Saeedloei, and Marina De Vos, editors, *Technical Communications of the 32nd International Conference on Logic Programming, ICLP 2016*, volume 52 of *OASIcs*, pages 2:1–2:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[44] Martin Gebser, Benjamin Kaufmann, and Torsten Schaub. Conflict-driven answer set solving: From theory to practice. *Artificial Intelligence*, 187:52–89, 2012.

[45] Paul Saikko, Carmine Dodaro, Mario Alviano, and Matti Järvisalo. A hybrid approach to optimization in answer set programming. In Michael Thielscher, Francesca Toni, and Frank Wolter, editors, *Proceedings of the 16th International Conference on Principles of Knowledge Representation and Reasoning, KR 2018*, pages 32–41. AAAI Press, 2018.

[46] Michael Rosemann and Jan vom Brocke. The six core elements of business process management. In Jan vom Brocke and Michael Rosemann, editors, *Handbook on Business Process Management 1: Introduction, Methods, and Information Systems*, International Handbooks on Information Systems, pages 105–122. Springer, 2 edition, 2015.

[47] Geary A. Rummler and Alan J. Ramias. A framework for defining and designing the structure of work. In Jan vom Brocke and Michael Rosemann, editors, *Handbook on Business Process Management 1: Introduction, Methods, and Information Systems*, International Handbooks on Information Systems, pages 81–104. Springer, 2 edition, 2015.

[48] Sana Bouajaja and Najoua Dridi. A survey on human resource allocation problem and its applications. *Operational Research*, 17(2):339–369, 2017.

[49] Luise Pufahl, Sven Ihde, Fabian Stiehle, Mathias Weske, and Ingo Weber. Automatic resource allocation in business processes: A systematic literature survey. *The Computing Research Repository (CoRR), arXiv*.

[50] Giray Havur and Cristina Cabanillas. History-aware dynamic process fragmentation for risk-aware resource allocation. In Hervé Panetto, Christophe Debruyne,

Martin Hepp, Dave Lewis, Claudio Agostino Ardagna, and Robert Meersman, editors, *Proceedings of OTM 2019 Conferences - Confederated International Conferences: CoopIS, ODBASE, C&TC 2019*.

[51] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Multi-shot ASP solving with clingo. *Theory Practice of Logic Programming*, 19(1):27–82, 2019.

Published in the Proceedings of the 5th International Symposium on Data-Driven Process Discovery and Analysis (SIMPDA 2015), pp. 1-27, Jan 2017, Springer LNBIP vol. 244

# A Framework for Safety-critical Process Management in Engineering Projects*

Saimir Bala[1], Cristina Cabanillas[1], Alois Haselböck[2], Giray Havur[1]
Jan Mendling[1], Axel Polleres[1], Simon Sperl[2], and Simon Steyskal[1,2]

[1]Vienna University of Economics and Business, Austria
{name.surname}@wu.ac.at
[2]Siemens AG Österreich, Corporate Technology, Vienna, Austria
{name.surname}@siemens.com

**Abstract.** Complex technical systems, industrial systems or infrastructure systems are rich of customizable features and raise high demands on quality and safety-critical aspects. To create complete, valid and reliable planning and customization process data for a product deployment, an overarching engineering process is crucial for the successful completion of a project. In this paper, we introduce a framework for process management in complex engineering projects which are subject to a large amount of constraints and make use of heterogeneous data sources. In addition, we propose solutions for the framework components and describe a proof-of-concept implementation of the framework as an extension of a well-known BPMS.

**Keywords:** Adaptation, compliance, engineering, process management, resource management, unstructured data

## 1 Introduction

Deployments of technical infrastructure products are a crucial part in the value-creation chain of production systems for large-scale infrastructure providers. Examples of a large-scale, complex engineering process in a distributed and heterogeneous environment are the construction of a railway system comprising electronic interlocking systems, European train control systems, operator terminals, railroad crossing systems, etc.; all of the systems are available in different versions using a variety of technologies. It is often necessary to offer, customize, integrate, and deliver a subset of these components for a particular customer project, e.g. the equipment of several train stations with an electronic switching unit in combination with a train control system based on radio communication for a local railway company. Configurators are engineering tools for planning and customizing a product. Each subsystem comes with its own, specialized engineering and verification tools. Therefore, configuring and combining these subsystem data to a coherent and consistent system has to follow a complex, collaborative process.

A challenge is the management and monitoring of such complex, yet mostly informally described, engineering processes that involve loosely integrated components,

* This work is funded by the Austrian Research Promotion Agency (FFG) under grant 845638 (SHAPE): http://ai.wu.ac.at/shape-project/

configurators and software systems [23]. Nowadays, many of the steps required (e.g. resource scheduling, document generation, compliance checking) tend to be done manually, which is error-prone and leads to high process execution times, hence potentially affecting costs in a negative way.

In this paper, we explore this domain and present a framework for process management in complex engineering processes that includes the formalization of human-centric process models, the integration of heterogeneous data sources, rule enforcement and compliance checking automation, and adaptability, among others. The framework has been defined from an industry scenario from the railway automation domain. Furthermore, we describe solutions to support the functionalities required by every framework component as well as a proof-of-concept implementation of the framework that can be integrated with an existing Business Process Management System (BPMS). The goal is to help to develop ICT support for more rigorous and verifiable process management.

The rest of the paper is organized as follows. Section 2 delves into the problem and derives system requirements. Section 3 presents the framework and solutions for its components. Section 4 describes a proof-of-concept implementation. Section 5 summarizes related work. Finally, Section 6 draws conclusions from this work and outlines ideas for future extensions.

## 2 Motivation

In the following, we describe an industry scenario that exemplifies the characteristics of complex engineering processes and define a set of system requirements for the challenges identified in it.

### 2.1 Industry Scenario

Activities to create complete, valid and reliable planning, and customization process data for a product deployment are part of an overarching engineering process that is of crucial importance for the success of a project in a distributed, heterogeneous environment. Fig. 1 depicts a generic engineering process for building a new infrastructure system in the railway automation domain modeled with Business Process Model and Notation (BPMN) [34].

The engineering process itself is represented in the pool *Railway automation unit* and comprises the building and testing of the system. The pool *Resource planning unit* as well as the activities depicted in gray represent a meta-process comprising scheduling activities that are performed in the background in order to enable the completion of the engineering process in compliance with a set of restrictions (temporal and logistics, among others) while making an appropriate use of the resources available. Resource allocation is of great importance to large-scale engineering processes in which a large variety of different resources, ranging from laboratories and specific hardware to engineers responsible for the correct execution of the process, are involved and unexpected situations may have critical consequences (e.g., delays resulting in unplanned higher costs).

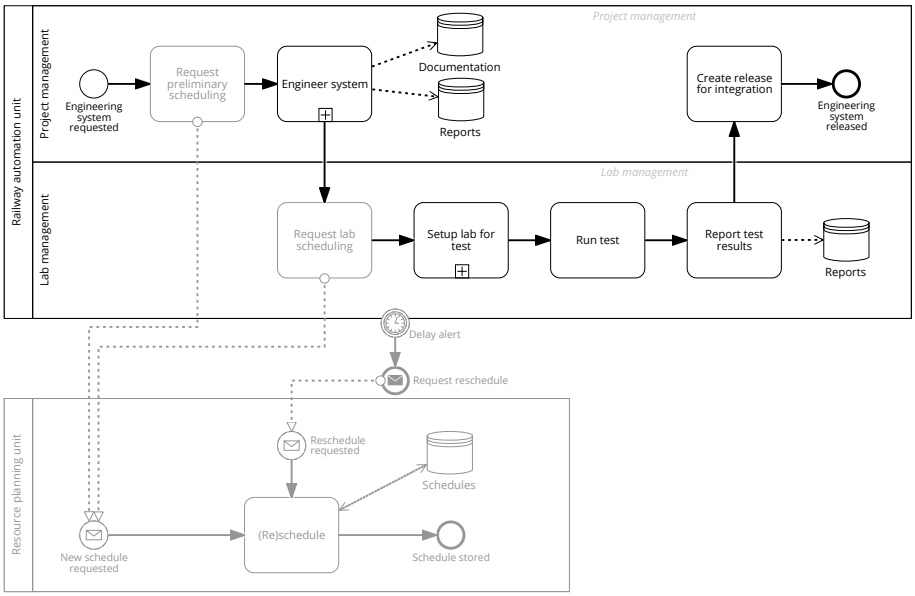



Fig. 1: Generic engineering process in the railway automation domain

Hence, the first step consists of scheduling the building of the system. Building the system is, in turn, a process composed of several activities (potentially operating on different levels of abstraction) each involving a large variety of different resources, data sources, and data formats used. Specifically, the customer provides input data in form of, e.g., XML documents representing railway topology plans, signal and route tables, etc., which are used by the engineers to configure the product. Typically, several configuration tools are involved in that process too, complemented by version control and documentation activities. The result is a set of data of various kinds and formats (i.e., XML, JSON, and alike) such as bill of material (BOM), assembly plans, software configuration parameters, and all other documents and information required for the testing, integration, and installation of the system. Additionally, we map all gathered data to a common extendable RDF model in order to make use of standard data integration and processing strategies from the Semantic Web (e.g., OWL, SPARQL, SHACL, etc.). The engineering project manager orchestrates and monitors these engineering tasks. Besides, further data is generated during the execution of the subprocess *Engineer system* in the form of, e.g., emails exchanged between the process participants.

Once the system is built, it must be tested before it is released for its use. That procedure takes place in laboratories and comprises two phases: the test setup and run phases. Like before, it is necessary to schedule these activities taking into consideration the setting and all the restrictions for the execution of the activities. The setting is the following: there are several space units distributed into two laboratories and several units of different types of hardware for conducting the testing. The employees of the organization involved in these activities are specialized in the execution of specific testing

phases for specific types of systems, i.e. there may be an engineer who can perform the setup for a system $S_1$ and the test execution for a system $S_2$, another engineer who cannot be involved in the testing of system $S_1$ but can perform the two activities for the system $S_2$, and so on. As for the restrictions, they are as follows: each task involved in these two phases requires a specific set of resources for its completion. In particular, the setup tasks usually require one employee working on one unit of a specific type of hardware in a laboratory, and the run activity usually requires several employees for its execution. Besides, a test can only be executed if the whole setup takes place in the same laboratory. In addition, for the scheduling it is necessary to take into account that other instances of the same or different processes might be under execution at the same time and they might share resources.

The setup and the run test activities will then be executed according to the plan. Similar to the engineering step, data comprising the results of the tests, emails, Version Control System (VCS) file updates and the like, is generated during the testing steps. Railway projects also generate other types of data which play a role in the running system, e.g., cut plans, signals form the tracks, actual user data of the system, etc. The latter can be useful when it comes to monitoring safety-critical processes during their execution. Given the great number of software tools involved in the process, our scenario focuses more on the software engineering aspect of the railway domain. Hence, we use VCS logs to track the evolution of the artifacts that are produced during such software engineering process. Nevertheless, it can be extended towards all kinds of artifacts that are stored in VCSs, such as the different versions of outputs from engineering tools.

When the testing of the system is finished, a final report is written and archived with the information generated containing the description of the test cases, test data, test results, and the outline of the findings. Responsible for the final version of this report is the testing project manager. Finally, the engineering project manager deploys a complete and tested version of the engineering system and the integration team takes over the installation of the product.

Note that unexpected situations may cause delays in the completion of any of the activities involved in the engineering process. It is important to detect such delays as soon as possible in order to properly schedule the use of resources and figure out when the process can be finished under the new circumstances. Therefore, rescheduling may be required at any point, involving all the aforementioned restrictions and possibly new ones.

### 2.2 Challenges and Technical Requirements

A number of issues are involved in the industry scenario previously described when it comes to automating its execution. From the analysis of the process description, the following challenges have been identified:

*Challenge 1: Integrated description of processes, constraints, resources and data.* Operating with processes like the one described before implies taking not only the order of execution of the process activities and behavioural constraints typically enforced in the process model into consideration, but also information related to other business processes perspectives, such as resources and data requirements, as well as regulations affecting, e.g., the use of these. Several formal languages are at hand for describing

processes [34], constraints [27], resources [11] and data (e.g. XML) separately. However, a challenge is to define all them in an integrated manner with a model that provides rich querying capabilities to support analysis automation, status monitoring or respectively, the verification of constraints and consistency.

Therefore, a system for automating processes like the engineering process would require *an integrated semantic model to describe and monitor processes, resources, constraints and data (RQ1).*

*Challenge 2: Integration and monitoring of structured and unstructured data.* To a high degree, engineering steps are the input for state changes of the process, often only visible as manipulation of data. Hence, a engineering process must also incorporate these data smoothly for monitoring control flow, version updates, data storage, and email notification. To this end, various types of systems have to be integrated including their structured (e.g., logs from tools, or databases) and unstructured data (e.g., by mail traffic, or ticketing systems). Up until now, these data are hardly integrated and are monitored mostly manually.

Therefore, techniques to gather relevant information from unstructured or semi-structured data sources, such as emails, VCS repositories and data from tools, and transform them into understandable data structures must be put in place, i.e. a system for automating processes like the engineering process would require *mechanisms to detect and extract structured from unstructured process data (RQ2).*

*Challenge 3: Documentation of safety-critical, human and data aspects and compliance checking.* Engineering projects have time-critical phases typically prone to sloppy documentation and reduced quality of results. Many of the process steps are required to be documented in prescribed ways by standards and regulations (e.g. SIL [6]). Considerable amount of time is spent in the manual documentation of process steps as well as in the integration of the documentation of separate modules for generating final reports. Furthermore, in such safety-critical environments the use of resources must be optimized and rules must be enforced and their fulfillment ensured. For instance, in our industry scenario we can observe a typical series of data management steps, including: check in a new version of a data file, inform the subsequent data engineer, confirm and document this step in the process engine, etc. The latter two steps could be done automatically once the process engine has detected the check-in into the VCS. This automation would also lead to a significant decrease of the overall execution time as well as a potential reduction in the number errors typically caused by human mistakes.

Therefore, a system for automating processes like the engineering process would require *a method for flexible document generation (RQ3)* as well as *reasoning mechanisms (RQ4)* and *monitoring capabilities (RQ5)* for automating resource allocation and compliance checking.

*Challenge 4: Be ready for changes.* Despite engineering process definitions are quite stable and might remain unchanged for a long time, an automatic monitoring and a thorough analysis of process models and executions may lead to the discovery of potential improvement points to make processes simpler and less error-prone. Similarly, changes in the schedule of activities and resources can be necessary at any time due to a number of reasons including delays or unexpected unavailability of resources, among others. Currently, all these adaptations require manual work and are prone to errors.

Consequently, methods to detect and deal with changes must be put in place. This might include the monitoring of process executions and the analysis of the generated execution data (e.g. with process mining techniques) to anticipate delays as well as the need of having available mechanisms capable of reallocating the resources according to changeable requirements and circumstances. Therefore, besides requirements RQ4 and RQ5, a system for automating processes like the engineering process would also require *adaptation procedures to react to changing conditions (RQ6)*.

*Challenge 5: Acceptance and human factors.* The overall process management needs to be set up in a non-obtrusive way, such that engineers executing the processes find it useful and easy to use. This is a specific challenge in safety-critical systems, which are developed with a tight timeline. It calls for a design that integrates existing tools and working styles instead of introducing new systems.

Therefore, the automation of processes like the engineering process would require *an integrated system (RQ7)* that provides all the functionality, which involves general features of a BPMS (e.g. process modeling and execution) extended to support the demands of safety-critical, human- and data-centric processes as described in our industry scenario.

## 3 Framework

We have designed a framework that provides the support required to address the challenges identified in complex engineering processes. It consists of a data model and five functional modules that interact with a BPMS, as depicted in Fig. 2 using the Fundamental Modeling Concepts (FMC) notation[1].

In order to support *RQ1*, a semantic model encompassing the various types of domain data that must be represented and manipulated must be defined. Hence, this model stores all static and dynamic data used by the BPMS and by the functional components. Specifically, these data include: process models and their instances, organizational data related to human resources, infrastructure data related to non-human resources, and constraints derived from regulations and norms (e.g. SIL) as well as further requirements related to the utilization of resources. The semantic model implicitly operates as a communication channel between the BPMS and all the functional modules and hence, all of them must have read&write access to the model.

Typical functionality of a BPMS include modeling and executing processes. Information about process instances is usually stored in event logs generally including, among others, temporal and resource information related to the execution of the process activities [45]. In addition to that structured information, as described in Section 2.1, several kinds of unstructured and semistructured data are generated during the execution of complex engineering processes, e.g. emails, VCS files and reports. All the data produced during process execution must be analyzed in order to detect anomalies (e.g., deviations from the expected behavior).

The *Process Miner* component of our framework tries to discover as much data relevant to the current state of a process execution as possible, performs the transformations required as specified by *RQ2*, and communicates the information extracted to the

[1] `http://www.fmc-modeling.org/`

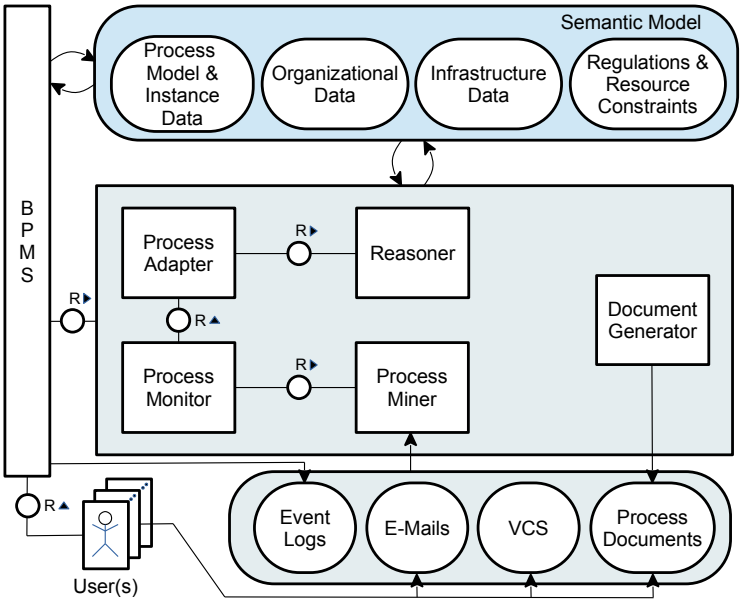



Fig. 2: Proposed framework for process management in complex engineering projects

*Process Monitor* (*RQ5*) periodically under request. In case the *Process Monitor* reveals a discrepancy between process instance data and the data discovered by the process miner (e.g., a delay), it informs the *Process Adapter* about the discrepancy. The *Process Adapter* analyzes the deviation and responds by proposing an adaptation solution to the BPMS in order to put the process back into a coherent and consistent state, as specified in *RQ6*. The adaptation may consist of small changes that can be performed directly on the BPMS side or, on the contrary, of complex recovery actions that may require reasoning functionalities. In the latter case, the *Reasoner* comes into play by, e.g., doing a new activity or resource scheduling according to the new domain conditions. Therefore, the *Reasoner* can be seen as a supportive component that helps the BPMS with typical activities, such as the scheduling of process activities, and the allocation of resources to those activities in accordance with resource constraints and regulations defined in the semantic model. This covers *RQ4*.

Finally, the *Document Generator* of the framework provides support for *RQ3* by helping to fill out the documents that must be generated as output of process activities. As mentioned before, this automation is expected to decrease reporting errors, especially in documents related to auditing.

The design of the framework as an extension of the functionality present in existing BPMSs attends to *RQ7* and hence, it intends to increase the acceptance by users familiar with Business Process Management (BPM).

In the following, we describe our solution for the implementation of the functionality provided by the most domain specific components of the framework.

### 3.1 Semantic Model

Aiming at automation, we believe that Semantic Web technologies provide the most appropriate means for (i) integrating and representing domain-specific (heterogeneous) knowledge in a consistent and coherent format, and (ii) querying and processing integrated knowledge.

Therefore, following the METHONTOLOGY approach [32], we have developed an engineering domain ontology [9] that integrated three different domains of interest relevant for our approach, namely: (i) engineering domain and organizational (i.e. resource-related) knowledge; (ii) business processes; and (iii) regulations and policies [41].

**Representing Infrastructural & Organizational Knowledge** One of the first steps for developing an ontology according to the METHONTOLOGY approach involves the definition of an *Ontology Requirements Specification Document* [43]. In order to address the requirements gathered throughout that process we decided to adopt parts of the organizational meta model described in [37] and enriched it with concepts for modeling teams [12] (cf. Fig. 3) for representing infrastructural & organizational knowledge. Using these two meta models presents an advantage. Specifically, the organizational meta model described in [37] has been used to design a language for defining resource assignment conditions in process models called Resource Assignment Language (RAL) [11]. As was shown in [14], that language can be seamlessly integrated in existing process modeling notations, such as BPMN, thus enriching the process models with expressive resource assignments that cover a variety of needs described by the creation patterns of the well-known workflow resource patterns [37]. Furthermore, a graphical notation was later designed with the same expressive power as RAL in order to help the modeler to define resource assignments in process models [10]. The meta model for teamwork assignment was also considered to develop an extension of RAL called RALTeam [12], which, however, lacks a graphical notation so far. Therefore, if support for these expressive notations were introduced in the BPMS, the ontology would support them at the same time as it supports less expressive means of assigning resources to process activities (e.g. based on organizational positions).

**Representing Business Processes** Driven by the requirements of our resource allocation approach (cf. Section 3.2), we decided to transform BPMN models into timed Petri nets [55] as an intermediary format for reasoning tasks (e.g., scheduling [28]) by using the transformation proposed in [20, 19], and store these Petri nets in our ontology. Note that the user only interacts with the BPMN model while using the system as we use the timed Petri net representation internally. There are several reasons for using Petri nets for process modeling [47], namely:

- *Clear and precise definition:* Semantics of the classical Petri net is defined formally.
- *Expressiveness:* The primitives needed to model a business process (e.g. routing constructs, choices, etc.) are supported.
- *Tool-independent:* Petri nets have mappings to/from different business modelling standards [31]. Moreover, this immunizes our ontology from changes in business modeling standards.

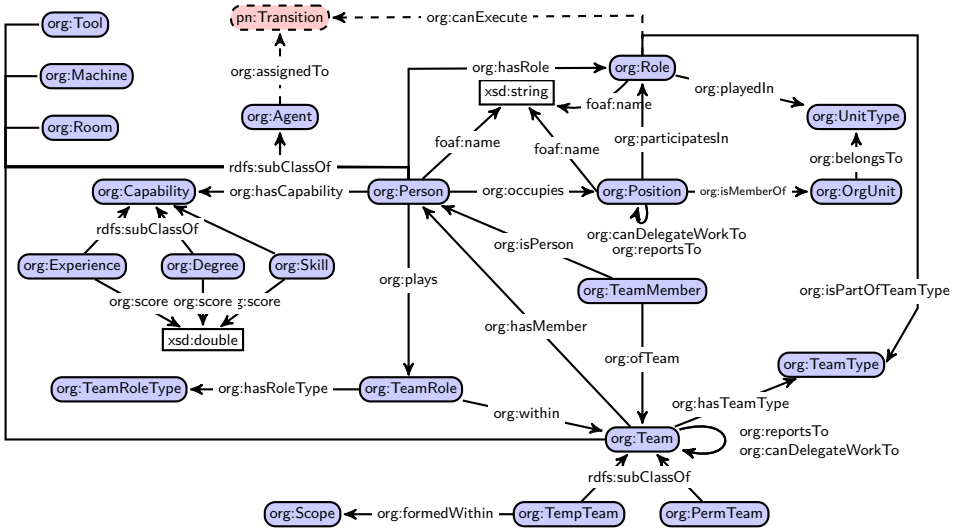



Fig. 3: Ontology for infrastructural & organizational knowledge.

For modeling Petri nets themselves we adopted selected concepts of the Petri Net Markup Language (PNML) [52] and represented them in terms of an RDFS ontology (cf. Fig. 4).

**Extracting and Specifying Compliance Rules** One of the most important aspects of dealing with safety-critical human- and data-centric processes is providing means for proving that business processes comply with relevant guidelines such as domain-specific norms and regulations, or workflow patterns. As illustrated in Fig. 5 and described in [33], establishing proper compliance checking functionalities typically requires to extract and interpret a set of *Compliance Objectives* from respective *Compliance Requirements* first, before those objectives are specified in terms of *Compliance Rules/Constraints* using an appropriate specification language (i.e. a language capable of representing all types of compliance rules/constraints relevant for the respective domain of interest). Specified compliance rules and constraints are then subsequently used by a monitoring/compliance checking engine for verifying correct and valid execution of business processes w.r.t. previously defined rules.

1. *Identifying Compliance Objectives:* Organizations have to deal with an increasing number of norms and regulations that stem from various compliance sources, such normative laws and requirements. In the railway domain processes have to be compliant with specific European Norms (i.e. 50126, 50128, and 50129). For example, EN50126 defines guidelines for managing Reliability, Availability, Maintainability, and Safety (RAMS) of safety-critical business processes, where we extracted objectives, requirements, deliverables, and validation activities defined in all phases of the RAMS lifecycle [24, 40].

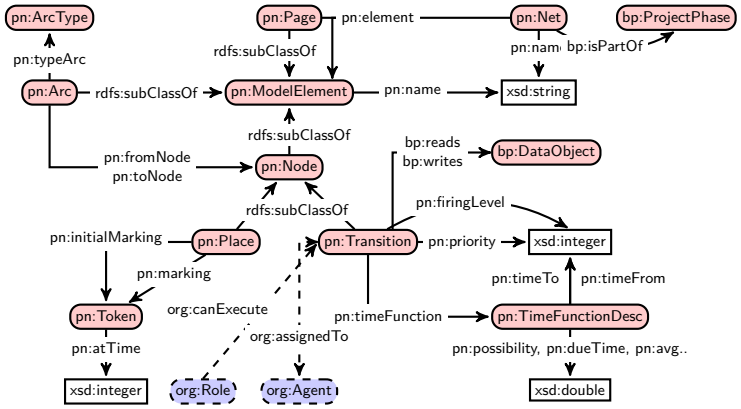



Fig. 4: Ontology for representing processes and process instances.

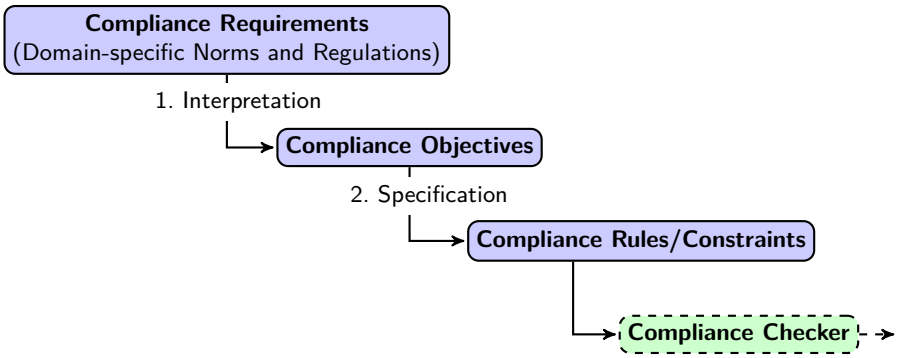



Fig. 5: General approach for business process compliance monitoring [33].

2. *Representing Compliance Rules and Constraints:* Since all process relevant data are stored in RDF, we plan to utilize recent advancements in the area of constraint checking for RDF, i.e. the Shapes Constraint Language (SHACL) [30] for representing and validating identified compliance objectives. Since constraints expressed in SHACL are internally mapped to corresponding SPARQL queries, we can further complement our own compliance constraints with already existing approaches for compliance checking using SPARQL such as [4]. Compliance constraints expressed in SHACL are tightly integrated with the underlying ontology and can be validated during both design time and runtime[2].

### 3.2 Reasoner

The *reasoner* module supports our framework on top of the engineering domain ontology in two folds: (i) by performing automated resource allocation described in the declarative

[2] For a more detailed introduction on utilizing SHACL for defining custom constraints, we refer the interested reader to [30].

formalism Answer Set Programming (ASP) [25], and (ii) by querying the ontology for compliance checking. We also looked at other declarative programming paradigms (e.g., CLP(FD)), and our initial findings confirm the advantages of using ASP [7, 21]. Some of these advantages are as follows:

- Compact, declarative and intuitive problem encoding
- Rapid prototyping and easy maintenance (e.g., no need to define heuristics)
- Complex reasoning modes (e.g., weight optimization)
- Ability to model effectively incomplete specifications
- Efficient solvers (e.g., *clingo*)

In the literature, ASP is preferable when the size of the problem does not explode the grounding of the program [21, 1]. We show that our resource allocation encoding in ASP is applicable to the problems of business processes at a real-world scale [28].

**Resource Allocation**

Resource allocation aims at scheduling activities of a business process and properly distributing available resources among scheduled activities. We address the problem of allocating the resources available to the activities in the running process instances in a time optimal way, i.e. process instances are completed in the minimum amount of time. Therefore, our resource allocation technique makes business process executions effective and efficient.

We encode the resource allocation problem in Answer Set Programming (ASP) [25], a declarative (logic programming style) paradigm. Its expressive representation language, efficient solvers, and ease of use facilitate implementation of combinatorial search and optimization problems (primarily *NP-hard*) such as resource allocation. Therefore, modifying, refining, and extending our resource allocation encoding is uncomplicated due to the strong declarative aspect of ASP. We use the ASP solver *clasp* [25] for our purpose as it has proved to be one of the most efficient implementations available [15]. Another complex reasoning extension supported in *clasp* are weight optimization statements [25] to indicate preferences between possible answer sets.

*Resources* are defined in the engineering domain ontology (the organizational data and the infrastructure data) where they are characterized by a *type* and can have one or more *attributes*. In particular, any resource type (e.g., org:Person in Fig. 3) is a subclass of org:Agent. The attributes are all of type rdf:Property. The organizational data consists of human resources, their attributes (e.g. their name, role(s), experience level, etc.) and current availabilities stored in the ontology. In the same fashion, infrastructure data represents material resources (i.e. tools, machines, rooms) and their availabilities. Resource allocation considers resources to be *discrete* and *cumulative*. Discrete resources are either fully available or fully busy/occupied. This applies to many types of resources, e.g. people, software or hardware. However, for certain types of infrastructure, availability can be partial at a specific point in time. For instance, a room's occupancy changes over time. Such a cumulative resource is hence characterized by its *dynamic* attribute (available space in the room) and it can be allocated to more than one activity at a time. Any statement in our ontology can be easily incorporated as the input of our problem

encoding [22]. The following example shows an excerpt of organizational data in the ontology and its equivalent in ASP.

```
# Organizational data
:glen a org:Person; foaf:name "Glen";
      org:occupies testeng.
:testeng a org:Position; foaf:name "Test Engineer";
         org:participatesIn labmng.
:labmng a org:Role; foaf:name "Lab Management".

% Equivalent ASP encoding
person(glen). name(glen,"Glen").
occupies(glen,testeng).
position(testeng). name(testeng,"Test Engineer").
participatesIn(testeng,labmng).
role(labmng). name(labmng,"Lab Management").
```

There are two main operations under resource allocation: *Allocation of resources* and *re-allocation of resources as adaptation*.

*Allocation of resources* deals with the assignment of resources and time intervals to the execution of process activities. It can be seen as a two-step definition of restrictions. First, the so-called *resource assignments* must be defined, i.e., the restrictions that determine which resources can be involved in the activities [11] according to their properties. The outcome of resource assignment is one or more *resource sets* with the set of resources that can be potentially allocated to an activity at run time. The second step assigns cardinality to the resource sets such that different settings can be described.

As mentioned in Section 3.1, there exist languages for assigning resource sets to process activities [11, 46, 42, 13]. However, cardinality is generally disregarded under the assumption that only one resource will be allocated to each process activity. This is a limitation of current BPMS, which we overcome in our proposed framework.

The main temporal aspect is determined by the expected duration of the activities. The duration can be predefined according to the type of activity or calculated from previous executions, usually taking the average duration as reference. This information can be included in the executable process model as a property of an activity (e.g. with BPMN [34]) or can be modelled externally. As for the variable activity durations depending of the resource allocation, three specificity levels can be distinguished:

- *Role-based duration*, i.e., a triple $(activity, role, duration)$ stating the (minimum/average) amount of time that it takes to the resources within a specific resource set (i.e., cardinality is disregarded) to execute instances of a certain activity.
- *Resource-based duration*, i.e., a triple $(activity, resource, duration)$ stating the (minimum/average) amount of time that it takes to a concrete resource to execute instances of a certain activity.
- *Aggregation-based duration*, i.e., a triple $(activity, group, duration)$ stating the (minimum/average) amount of time that it takes to a specific group to execute instances of a certain activity. In this paper, we use $group$ to refer to a set of human resources that work together in the completion of a work item, i.e., cardinality is considered. Therefore, a $group$ might be composed of resources from different roles

which may not necessarily share a specific role-based duration. An aggregation function must be implemented in order to derive the most appropriate duration for an activity when a group is allocated to it. The definition of that function is up to the organization.

Given (i) a process model and its instance data; (ii) organizational and infrastructure data; (iii) resource requirements, i.e. the characteristics of the resources that are involved in each activity to be allocated (e.g. roles or skills); (iv) temporal requirements; and (v) regulations such as access-control constraints [11], i.e. *separation of duties (SoD)* and *binding of duties (BoD)*, the ASP solver finds an optimal allocation. The aforementioned functionalities and the entire associated ASP encoding are detailed in [26].

While executing the process instance, changes may be introduced to input used for allocation. For instance, organizational data may change in case of absence, regulations may be modified or simply execution of activities may delay. In some cases, such a change in the ontology directly affects a running process instance, and therefore, the process monitor informs the process adapter. The process adapter may decide that the allocation should be performed again. *Adaptive re-allocation* is a key functionality in this scenario and it is indispensable for safety-critical, human- and data- centric process management.

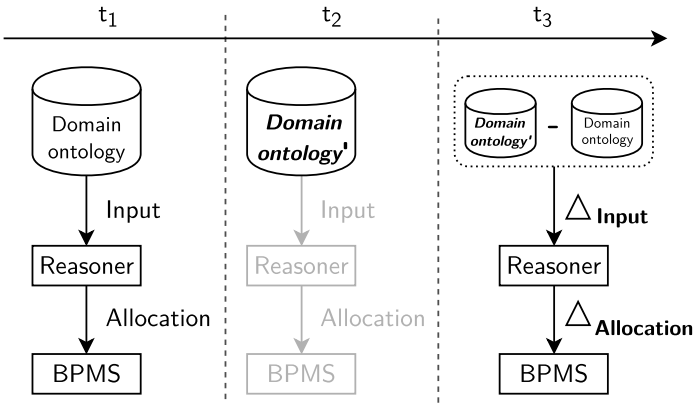



Fig. 6: Adaptive re-allocation timeline

Fig. 6 shows this scenario in three consecutive time steps: After allocating resources to a process instance at $t_1$, some changes are introduced at $t_2$ that interfere with the original allocation, and hence, an adaptive reallocation is performed at $t_3$. The reasoner computes a delta allocation, i.e. the original allocation is preserved as much as possible. Therefore, some activities might be rescheduled, and others might be shifted and/or reallocated to some different resources in a *minimal* fashion.

**Compliance Checking** As mentioned previously, we define compliance constraints over business processes using SHACL. In order to do so, we translate each compliance

objective to a corresponding SPARQL query first, before embedding it in a respective constraint component, which itself can then be integrated into the ontology [40].

### 3.3 Process Monitor and Process Adapter

Changes and deviations to the processes may occur during execution. For instance, new rules and regulations may require the process to operate differently. We want our framework to be able to handle these unexpected events.

The process data and the evidence from the process miner are compared by the process monitor for detecting deviations. The main idea is that the process adapter is informed about the deviations, therefore it minimizes the impact of these deviations in the running process instances by offering a recovery strategy. The process monitor and process adapter address the requirements *RQ5* and *RQ6*, respectively. The process monitor is able to run several algorithms for monitoring both the process behaviour and the process compliance to rules and regulations. This is performed by checking the current process constraints against the data from our semantic model.

The process adapter is in charge of handling exceptions that arise from the process monitor. This component acts in two different ways: Either it *i)* corrects process behaviour with minimal intervention, or, in case a more complex adaptation is required, *ii)* it stops the process and notifies the reasoner for planning an adaptation.

### 3.4 Process Miner

Traditional process mining algorithms [45, 49] are able to give valuable insights into the different perspectives of a business process. Process models inferred from log files can further be analyzed for bottlenecks, performance, deviations from the expected behaviour, compliance with rules and regulations, etc. Regardless of the perspective they aim at mining as well as the type of process modeling notation used to represent the outcome (declarative versus imperative process mining), all these process mining algorithms require properly structured data. Specifically, they must comply with the XES [51] meta model. Any of the existing process mining techniques is a candidate to be used for the implementation of the *Process Miner* component in regard to the functionality related to traditional process mining and the decision should be made according to the specific characteristics desired.

However, the biggest challenge of this component in our framework is to deal with unstructured and semistructured data generated in the execution of the process activities, generally in the form of VCS files and emails. Although it is hard to mine process models out of such unstructured or semistructured data, some approaches can be used to obtain valuable insights on them. Specifically, [29] allow for transforming semi-structured VCS logs to process activities which can be mined by classic mining algorithms. Poncin et al.[36] developed the FRASR framework, which is enables to answers engineering questions by applying process to software repositories. The challenge here is to identify the relevant events for the files, from a process mining point of view. Di Ciccio et al. [18] propose the MAILOFMINE approach to discover artful processes laying underneath email collections. Bala et al. [2] adopt a visualization approach by mining project Gantt charts from VCS logs.

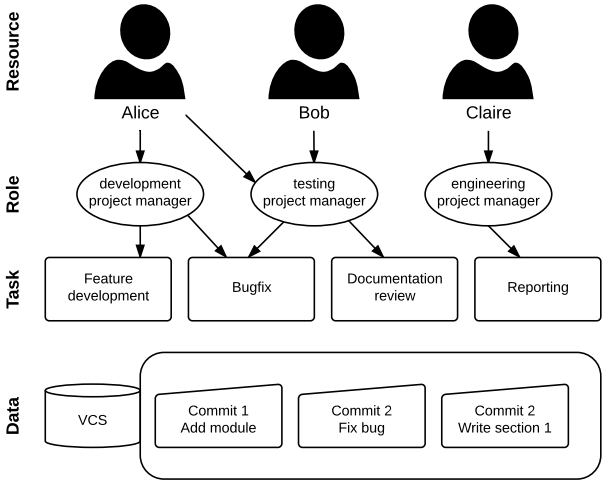



Fig. 7: Software project and resources

Driven by the fact that complex engineering process like the industry scenario described in Section 2.1 are resource-intensive, we have developed a novel approach to extract organizational roles from VCS repositories. VCSs have both structured and unstructured data. On the one hand, they explicitly provide information about the user who performed changes in some file(s) and the time at which she committed the new version(s). On the other hand, they have a textual part typically carrying a comment that explains the changes performed on the file(s). Note that these kind of data are similar to data from email. In fact, both emails and git comments have in common information about the user, the timestamp, and a textual description. Moreover, we use an ontology (cf. Section 3.1) to store mining insights. That is, we allow for the integration of all types of data that can be represented in RDF, including data coming from engineering tools. There are indeed tools in SVN [35] or Git [44] that allow for sending user commits as emails. Therefore, discovering roles out of such data (and especially when the outcome is combined with the result of mining activities) might help the *Process Monitor* to identify potential deviations regarding the resources that have actually performed specific tasks or manipulated certain information. Hence, it contributes to compliance checking.

Let us see an example of users who use VCS to collaborate on a software development project. Fig. 7 shows a setting with three users named Alice, Bob and Claire. Alice is a development project manager. She works with her colleagues Bob and Claire. Alice is mainly responsible for *feature development*, but she is also involved in *testing* project-management team. Her tasks include the *development* of new features and fixing of related bugs. In her first *commit* she adds a new message where she describes her work. The message to describe her changes is “added new module to demo”. In the first row of Table 1, identified by *commit id 1*, Alice’s change is reported. Bob is part of the *testing* project-management team. His task is to ensure that the code submitted by the development team complies with existing standards and contains no errors. He discovers

and fixes some minor bugs in Alice's code and informs Alice on further work needed on the analyzed features. Meanwhile, he commits his changes with *commit id 2* and comments "Modified the setup interface". Consequently, Alice reworks her code and commits a new version as reported in row 3 of Table 1, commenting her work with "Update application interface". As an *engineering project manager*, Claire takes over and starts to work on the documentation. She commits part of her work as in row 4. As the project continues, the work is accordingly stored in the log as shown in the table.

| Id | User | Timestamp | Changed | Comment |
|---|---|---|---|---|
| 1 | Alice | 2014-10-12 13:29:09 | Demo.java rule.txt | Added new module to demo and updated rules |
| 2 | Bob | 2014-11-01 18:16:52 | Setup.exe | Modified the setup interface |
| 3 | Alice | 2015-06-14 09:13:14 | Demo.java | Update the application interface |
| 4 | Claire | 2015-07-12 15:05:43 | graph.svg todo.doc | Define initial process diagram & listed remaining tasks |
| ... | ... | ... | ... | ... |

Table 1: Example of a VCS log

Our approach leverages both on the file types and the comments of the users. We devise an algorithm that classifies users into a set of roles. For that purpose, we approach the role discovery problem as a classification problem. We define two methods: one based on user clustering and one based on commit clustering. A prior step for this is the feature selection. By looking at the commit data, we identify the following features:

- Total number of commits.
- Timeframe between the first and the last commit of a user (i.e. the time he has been working on the project).
- Commit frequency: total number of commits divided by the time frame.
- Commit message length: average number words in the commit comment.
- Keyword count: how often determinate words like "test" or "fix" are used
- Number of files changed.
- Affected file types: how often a file with a certain format (e.g., *.java, *.html) are modified by a user, relative to the total number of modified files

Then, we use the features for two machine learning algorithms. In the first approach we iterate through the users and cluster them using the k-means algorithm. Consequently we build classification models using decision trees. We then train three different datasets individually and cross validate the results. The second approach starts from the commits. The main idea here is that we do not want to assign users to a specific category. Rather, we allow for users having multiple roles and classify their contribution in each commit. We build user profiles that account for fractions of contributions of each user to the different classes. Classes used in the classification for the example described above would be: *Test, Development, Web, Backend, Maintenance, Refactor, Documentation, Design, Build, Data, Tool, Addition, Removal, vcsManagement, Automated, Merge*.

Each commit is classified into one of the classes according to its features. Users who committed can be then classified by their commits. The classification can be done both manually and automatically: *i)* rules can be manually inferred by looking for similarities between users with the same role; or *ii)* an automated classification can be performed by using machine learning algorithms. For example, decision trees can be used for an automated classification. In this case the *commit* type percentages are used as features and the manually assigned roles as classes. As a further step, the resulting decision tree models from the different datasets can be cross-validated. The complete approach and its evaluation can be found in [8].

### 3.5 Document Generator

Safety-critical engineering systems require well-documented process steps. Engineers are in charge of clearly describing their tasks in such a way that it is possible to audit their work. These documents are often manually created. This has at least four drawbacks. First, their creation is laborious. Second, it is error-prone and misaligned in terms of language. Third, it is described at different levels of granularity. Fourth, it is difficult to process and audit afterwards.

A simple example is the following. Engineers need to work on a specific task and use predefined tools. Their tasks and their version tools are specified at the beginning of the project and must be consistent during the project's lifetime. Tool versions must usually be filled in the documentation generated, e.g. in reports. In a big engineering project, tools can be numerous and their versions are far from being user-friendly. This makes the risk of human mistakes very likely.

To assist engineering project managers in producing audible documentations, we have developed a customizable approach for partially automating document generation. In particular, it is able to fill in trivial information (e.g., tool version, user name, task to which the user is assigned, etc) into word processor documents. Our document generation technique is based on templates. These templates consist of evolving documents and are automatically filled in during the workflow, and therefore enable flexible process verification. Our approach generates standard documents which are compatible with predefined word processor programs and can be opened and edited by them.

Document generation comprises four steps, depicted in Figure 8 and explained next. A mapping function is first defined from a process activity to a document which is generated as its outcome. Afterwards, an interpreter is defined which is in charge of filtering the relevant process activities and variables. Process variables are used by the BPMS during the execution of the process. Examples of process variables can be the name of the user that is currently assigned to one activity, the name of the running process instance, and everything that adds data to the executing process in the BPMS. The interpreter is not strictly bound to a particular process nor to a particular template, and is defined externally. This supports changes both in the process and in the template. The writing in the document is triggered by a listener. A listener waits for activity events. As soon as an activity is submitted, the interpreter and the mapping function work together to generate *(variable key,value)* pairs in the document template. This is run iteratively on the document until all the trivial data is filled in.

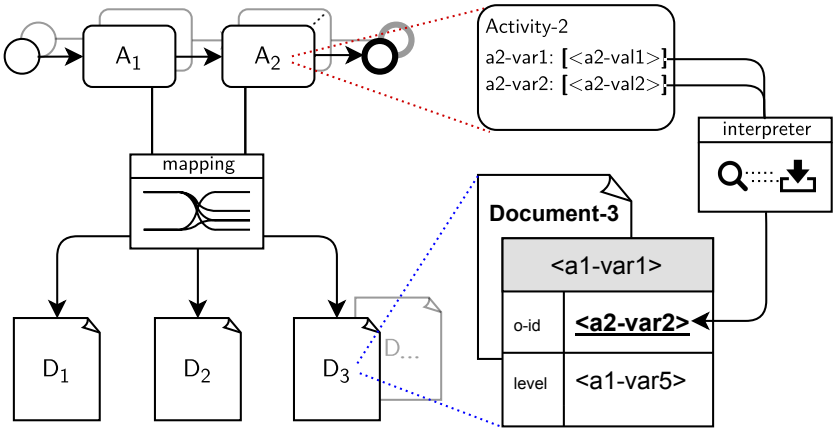



Fig. 8: Implementation showing how form values (process variables) from Camunda could look like in a generated document

## 4 Proof-of-concept Implementation

As a proof of concept we have implemented the main components of the framework discussed in Fig. 2. In this prototype, we aim to bring together functionality from reasoning, process mining and document generation. Fig. 9 shows the software architecture that we use. It considers four main components which interoperate during the execution of a process activity.

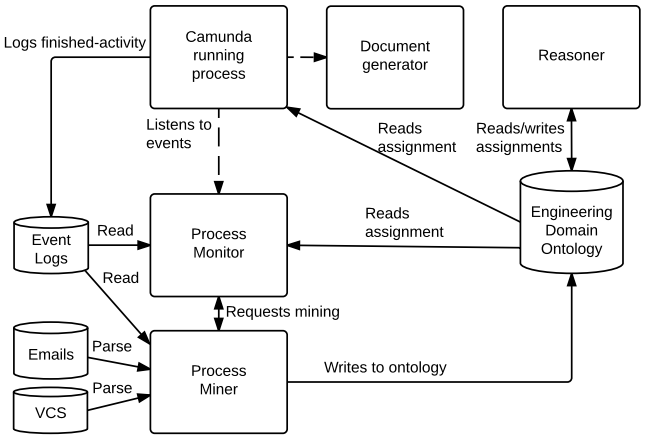



Fig. 9: Software architecture of the prototype

Here, we describe the main components of the architecture and their interactions.

**Camunda running process.** We use the Camunda BPM engine as our BPMS. Camunda is an open source platform that allows for defining new components and for interacting with its APIs in a custom way. All the process instances that run into Camunda and their data are stored in log files. Camunda uses two main databases to store its logs: *i)* a database for processes that are currently executing; and *ii)* a database for historical information. These two databases can be queried through provided Java or REST APIs. Results are returned as either a set of Plain Old Java Objects (POJOs) or in the JSON format, respectively.

Before an activity starts to run, it first fetches the ontology which contains the set of assignments from existing resources to activities. Consecutively, a resource is assigned to the activity and thus can appear on their task list. When the resources complete their tasks, an event is triggered. This event is listened by the process miner and the document generator components, who can react accordingly. At the same time, the event is stored into the Camunda database of the running instances. Both the running processes database and the history database record similarly-structured data. Furthermore, they can be accessed using the same technology, i.e. the Camunda REST APIs. Hence, we abstract both these databases as a single database in Figure 9 and denote it as *Event Logs*.

**Reasoner.** The reasoner module is implemented as a Java application connected to the Camunda process engine as an asynchronous service. We use Sesame, an open source framework for creating, parsing, storing, inferencing and querying over our ontology data. With respect to the request, the reasoner either performs resource allocation (cf. Figure 10) by first translating the RDF data into the ASP language, solving the problem instance using the ASP solver *clasp*, and then writing the allocation results back to the triple store; or it validates all contained SHACL constraints and returns potential violation result back to the process engine.

**Process Monitor.** This component is in charge querying the status of the running processes in Camunda. In case a deviation occurs, for example, a process instance cannot be completed within the assigned schedule, the process monitor must signal out the anomaly. The process adaptation module can use this output to learn the status of the system and subsequently apply an adaptation. This component is implemented as a web client that can read execution logs through the Camunda REST API. Results are returned in the JSON format which are then parsed into POJOs and can be processed by customized monitoring algorithms. In this case the communication happens through periodical queries to the database. An alternative to this is to implement an activity listener that notifies the process monitor whenever a task is completed.

**Miner.** The miner is in charge of running a number of mining algorithms on the logs from Camunda and from VCSs. Emails and commit messages can also be analysed by using the approaches discussed in [8]. This component is implemented as a web service, which can be called by the process monitor in order to understand how the activities being monitored have performed in the past. Mining algorithms can give new insights into the processes, like for instance actual execution times and several performance indicators of the process. This can contribute to the domain knowledge. Thus, they are stored again into the ontology as RDF.

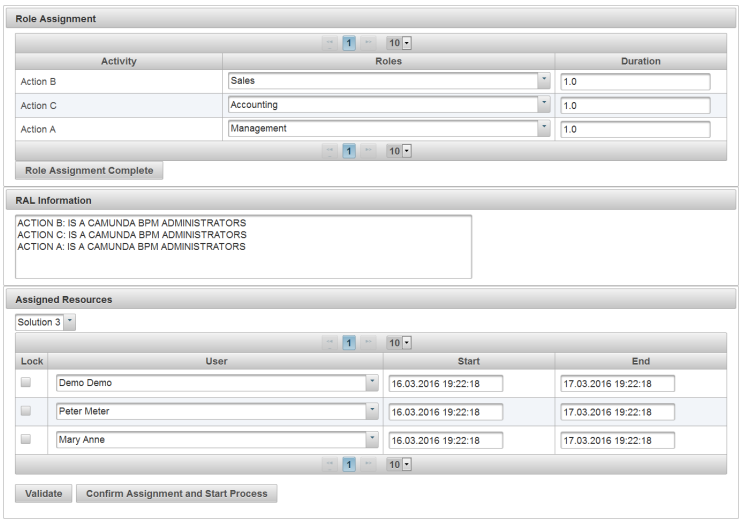



Fig. 10: Resource allocation interface

**Document generator.** The document generator is in charge of listening to activity submissions and of collecting information from them with the final goal of creating textual documents. This component uses customizable event handlers to process changes of process variables and forms compiled by the users. It is implemented in Java and can be imported as a Java library into several other modules that require document generation from events.

### 4.1 Limitations

The architecture is currently under implementation. The components have been only individually tested. There is the need for a comprehensive software solution that integrates the single software components into one.

**SQL console for querying Camunda logs.** We are developing a tool for process monitoring. This tool will allow for SQL-like queries on top of Camunda logs. The approach involves mapping Camundas database schema to RXES [50]. In addition to this we are also developing a tool that can map from RXES to XES [51] and we plan to use this tool with the approach from [38] in order to make it fully compatible with the RXES standard.

**Process adaptation.** The process adaptation module that we describe in the framework is yet to be implemented. This module will be developed as an intermediate component between the process monitor and the Camunda engine. It will act as a middle layer that is able to correct slight deviations in the running process, without stopping the workflow. Deviations that are not adjustable may occur. In this case, this component will communicate the need for a schedule to the reasoner.

**Connection to ontology.** Our ontology is currently under improvement. We are planning to complete it with all the data from the engineering domain ontology (cf. Figure 2). Furthermore, its connections to the various components are yet to be implemented.

**User interfaces.** We support for mining and monitoring techniques whose results are models that are generated out of data. User interfaces to visualize these data are required in order to easy the understanding of the mining results. Analogously, we plan to provide a fully fledged user interface for the reasoner component.

## 5 Related Work

The existing work on similar frameworks are from safety management [53, 5, 16], and decision support domains [54, 17]. To best of our knowledge, there is no framework addressing all the seven requirements (cf. Section 2) that we identify. Therefore, we elaborate on the supporting literature.

Bowen and Stavridou [5] detail the standards concerned with the development of safety-critical systems, and the software in such systems. They identify the challenges of safety-critical computer systems, define the measures for the correctness of such systems and its relevance to several industrial application areas of, e.g. formal methods in railway systems, which is crucial for rigorous and coherent process management.

De Medeiros et al. [17] investigate the core building blocks necessary to enable semantic process mining techniques/tools. They conclude that semantic process mining techniques improve the conventional ones by using three elements, i.e., ontologies, model references from elements in logs/models to concepts in ontologies, and reasoners. Our framework supports such a high-level semantic analysis through our integrated semantic model and the reasoner module.

Wilke et al. [53] describe a framework for a holistic risk assessment in airport operations. They focus on coordination and cooperation of various actors through a process model derived in BPM, which helps determination of causal factors underlying operation failures, and detection and evaluation of unexpected changes. The holistic consideration of operations handling rules and regulations of their particular domain serves for ensuring compliance. Daramola et al. [16] describe the use of ontologies in a scenario requiring identification of security threats and recommendation of defence actions. Their approach not only help the quick discovery of hidden security threats but also recommend appropriate countermeasures via their semantic framework. By following this approach, they minimize the human effort and enable the formulation of requirements in a consistent way. In our framework we similarly monitor our ontology for compliance checking by querying the ontology via the queries derived from regulations.

Van der Aalst [48] introduced a Petri net based scheduling approach to show that the Petri net formalism can be used to model activities, resources and temporal constraints with non-cyclic processes. Several attempts have also been done to implement the problem as a constraint satisfaction problem. For instance, Senkul and Toroslu [39] developed an architecture to specify resource allocation constraints and a Constraint Programming (CP) approach to schedule a workflow according to the constraints defined for the tasks. Our framework addresses resource allocation via the reasoner module using ASP.

Zahoransky et al. [54] investigate operational resilience of process management. Their approach is proposed as a complementary approach to risk-aware BPM systems, which focuses on detecting the resilience properties of processes based on measures by mining process-logs for decision support to increase process resilience, and therefore provide flexibility. This approach enables agility in run-time and provides a solid foundation for process execution reliability. We address these aspects in our integrated system via the process miner and the process adapter.

## 6 Conclusions and Future Work

In this paper we have explored challenges of safety-critical human- and data- centric process management in engineering projects which are subject to a large amount of regulations and restrictions, i.e. temporal, resource-related and logistical restrictions, as described in the industry scenario. Our proposed framework addresses all the requirements derived from those challenges upon the general functionality of a BPMS, e.g. process adaptation, resource allocation, document generation and compliance checking.

This work is developed in cooperation with SIEMENS Austria who will be the primary user of the developed system. Our first proof of concept is implemented[3]. Next steps also involve putting in place adaptation mechanisms, implementing and integrating all the components into Camunda, and conducting a thorough evaluation of the implemented system w.r.t. real data from the railway domain.

## References


[1] Markus Aschinger, Conrad Drescher, Gerhard Friedrich, Georg Gottlob, Peter Jeavons, Anna Ryabokon, and Evgenij Thorstensen. Optimization methods for the partner units problem. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 4–19. Springer, 2011.

[2] Saimir Bala, Cristina Cabanillas, Jan Mendling, Andreas Rogge-Solti, and Axel Polleres. Mining Project-Oriented Business Processes. In *BPM*, pages 425–440, 2015.

[3] Saimir Bala, Giray Havur, Simon Sperl, Simon Steyskal, Alois Haselböck, Jan Mendling, and Axel Polleres. SHAPEworks: A BPMS Extension for Complex Process Management. In *BPM Demos*, To appear, 2016.

[4] Khalil Riad Bouzidi, Catherine Faron-Zucker, Bruno Fies, and Nhan Le Thanh. An ontological approach for modeling technical standards for compliance checking. In *Web Reasoning and Rule Systems*, pages 244–249. Springer, 2011.

[5] Jonathan Bowen and Victoria Stavridou. Safety-critical systems, formal methods and standards. *Software Engineering Journal*, 8(4):189–209, 1993.

[6] M. Bozzano and A. Villafiorita. *Design and safety assessment of critical systems*. CRC Press Taylor & Francis Group, 2010.

[7] Gerhard Brewka, Thomas Eiter, and Mirosław Truszczyński. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2011.

[8] Cristina Cabanillas, Saimir Bala, Jan Mendling, and Axel Polleres. Combined method for mining and extracting processes, related events and compliance rules from unstructured data. Technical report, WU Vienna, 2016.

[9] Cristina Cabanillas, Alois Haselböck, Jan Mendling, Axel Polleres, Simon Sperl, and Simon Steyskal. Engineering Domain Ontology. SHAPE project deliverable.

[10] Cristina Cabanillas, David Knuplesch, Manuel Resinas, Manfred Reichert, Jan Mendling, and Antonio Ruiz-Cortés. RALph: A Graphical Notation for Resource Assignments in Business Processes. In *CAiSE*, volume 9097, pages 53–68. Springer, 2015.

[11] Cristina Cabanillas, Manuel Resinas, Adela del Río-Ortega, and Antonio Ruiz-Cortés. Specification and Automated Design-Time Analysis of the Business Process Human Resource Perspective. *Inf. Syst.*, 52:55–82, 2015.

[12] Cristina Cabanillas, Manuel Resinas, Jan Mendling, and Antonio Ruiz Cortés. Automated team selection and compliance checking in business processes. In *Proceedings of the 2015 International Conference on Software and System Process, ICSSP 2015, Tallinn, Estonia, August 24 - 26, 2015*, pages 42–51, 2015.

[13] Cristina Cabanillas, Manuel Resinas, Jan Mendling, and Antonio Ruiz Cortés. Automated team selection and compliance checking in business processes. In *ICSSP*, pages 42–51, 2015.

[14] Cristina Cabanillas, Manuel Resinas, and Antonio Ruiz-Cortés. RAL: A High-Level User-Oriented Resource Assignment Language for Business Processes. In *Business Process Management Workshops (BPD'11)*, pages 50–61, 2011.

[15] Francesco Calimeri, Martin Gebser, Marco Maratea, and Francesco Ricca. Design and results of the fifth answer set programming competition. *Artificial Intelligence*, 231, 2016.

[16] Olawande Daramola, Guttorm Sindre, and Thomas Moser. A tool-based semantic framework for security requirements specification. *J. UCS*, 19(13):1940–1962, 2013.

[17] Ana Karla Alves de Medeiros, Wil Van der Aalst, and Carlos Pedrinaci. Semantic process mining tools: core building blocks. 2008.

[18] Claudio Di Ciccio, Massimo Mecella, Monica Scannapieco, Diego Zardetto, and Tiziana Catarci. MailOfMine - Analyzing mail messages for mining artful collaborative processes. *Lect. Notes Bus. Inf. Process.*, 116 LNBIP:55–81, 2012.

[19] Remco M Dijkman, Marlon Dumas, and Chun Ouyang. Formal semantics and analysis of BPMN process models using Petri nets. Technical Report 7115, Queensland University of Technology, 2007.

[20] Remco M. Dijkman, Marlon Dumas, and Chun Ouyang. Semantics and analysis of business process models in BPMN. *Information & Software Technology*, 50(12):1281–1294, 2008.

[21] Agostino Dovier, Andrea Formisano, and Enrico Pontelli. A comparison of clp (fd) and asp solutions to np-complete problems. In *Logic Programming*, pages 67–82. Springer, 2005.

[22] Thomas Eiter, Giovambattista Ianni, Thomas Krennwallner, and Axel Polleres. Rules and Ontologies for the Semantic Web. In *Reasoning Web 2008*, volume 5224, pages 1–53. San Servolo Island, Venice, Italy, 2008.

[23] Gerhard Fleischanderl, Gerhard E. Friedrich, Alois Haselböck, Herwig Schreiner, and Markus Stumptner. Configuring Large Systems Using Generative Constraint Satisfaction. *IEEE Intelligent Systems*, 13(4):59–68, 1998.

[24] Julia Fuchsbauer. How to manage Processes according to the European Norm 50126 (EN 50126). Bachelor thesis, 2015.

[25] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. *Answer Set Solving in Practice*. Morgan & Claypool Publishers, 2012.

[26] Jan Mendling Axel Polleres Giray Havur, Cristina Cabanillas. Resource and data management service architecture. SHAPE project deliverable.

[27] Guido Governatori and Shazia Sadiq. The Journey to Business Process Compliance. In *Handbook of Research on BPM*, pages 426–454. IGI Global, 2009.

[28] Giray Havur, Cristina Cabanillas, Jan Mendling, and Axel Polleres. Automated Resource Allocation in Business Processes with Answer Set Programming. In *BPM Workshops (BPI)*, page In press, 2015.

[29] Ekkart Kindler, Vladimir Rubin, and Wilhelm Schäfer. Activity Mining for Discovering Software Process Models. *Softw. Eng.*, 79:175–180, 2006.

[30] Holger Knublauch and Arthur Ryman. Shapes Constraint Language (SHACL). Working Draft (work in progress), W3C, 2016. `https://www.w3.org/TR/shacl/`.

[31] Niels Lohmann, Eric Verbeek, and Remco Dijkman. Petri Net Transformations for Business Processes - A Survey. *Transactions on Petri Nets and Other Models of Concurrency II*, 2:46–63, 2009.

[32] Mariano F. Lopez, Asuncion G. Perez, and Natalia Juristo. METHONTOLOGY: from Ontological Art towards Ontological Engineering. In *AAAI97 Symposium*, pages 33–40, 1997.

[33] Linh Thao Ly, Fabrizio M. Maggi, Marco Montali, Stefanie Rinderle-Ma, and W.M.P. van der Aalst. Compliance monitoring in business processes: Functionalities, application, and tool-support. 54:209–234, March 2015.

[34] OMG. BPMN 2.0. Recommendation, OMG, 2011.

[35] C Michael Pilato, Ben Collins-Sussman, and Brian W Fitzpatrick. *Version control with subversion*. " O'Reilly Media, Inc.", 2008.

[36] Wouter Poncin, Alexander Serebrenik, and Mark Van Den Brand. Process Mining Software Repositories. *2011 15th Eur. Conf. Softw. Maint. Reengineering*, pages 5–14, 2011.

[37] Nick Russell, Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and David Edmond. Workflow Resource Patterns: Identification, Representation and Tool Support. In *CAiSE*, pages 216–232, 2005.

[38] Stefan Schönig, Cristina Cabanillas, Stefan Jablonski, and Jan Mendling. Mining the Organisational Perspective in Agile Business Processes. In *BPMDS*, pages 37–52, 2015.

[39] Pinar Senkul and Ismail H. Toroslu. An Architecture for Workflow Scheduling Under Resource Allocation Constraints. *Inf. Syst.*, 30(5):399–422, July 2005.

[40] Simon Steyskal. Engineering Domain Ontology. Project deliverable, Siemens, 2016.

[41] Simon Steyskal and Axel Polleres. Defining expressive access policies for linked data using the ODRL ontology 2.0. In *SEMANTICS 2014*, pages 20–23, 2014.

[42] L. J. R. Stroppi, O. Chiotti, and P. D. Villarreal. A BPMN 2.0 Extension to Define the Resource Perspective of Business Process Models. In *CIbS'11*, 2011.

[43] Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, and Boris Villazón-Terrazas. How to write and use the Ontology Requirements Specification Document. In *On the move to meaningful internet systems: OTM 2009*, pages 966–982. Springer, 2009.

[44] Linus Torvalds and Junio Hamano. Git: Fast version control system. *URL http//git-scm. com*, 2010.

[45] Wil van der Aalst. *Process mining: discovery, conformance and enhancement of business processes*. Springer-Verlag Berlin Heidelberg, 2011.

[46] Wil M. P. van der Aalst and Arthur H. M. ter Hofstede. YAWL: Yet Another Workflow Language. *Inf. Syst.*, 30(4):245–275, 2005.

[47] Wil MP Van der Aalst. The application of petri nets to workflow management. *Journal of circuits, systems, and computers*, 8(01):21–66, 1998.

[48] W.M.P. van der Aalst. Petri net based scheduling. *Operations-Research-Spektrum*, 18(4):219–229, 1996.

[49] B. F. Van Dongen, A. K A De Medeiros, H. M W Verbeek, A. J M M Weijters, and W. M P Van Der Aalst. The ProM framework: A new era in process mining tool support. In *Lect. Notes Comput. Sci.*, volume 3536, pages 444–454. Springer, 2005.

[50] B F van Dongen and Sh Shabani. Relational XES : Data Management for Process Mining. *BPM Cent. Rep. BPM-15-02*, 2015.

[51] H. M W Verbeek, Joos C A M Buijs, Boudewijn F. Van Dongen, and Wil M P Van Der Aalst. XES, XESame, and ProM 6. In *Lect. Notes Bus. Inf. Process.*, volume 72 LNBIP, pages 60–75. Springer, 2011.

[52] Michael Weber and Ekkart Kindler. The petri net markup language. In Hartmut Ehrig, Wolfgang Reisig, Grzegorz Rozenberg, and Herbert Weber, editors, *Petri Net Technology for Communication-Based Systems*, volume 2472 of *Lecture Notes in Computer Science*, pages 124–144. Springer, 2003.

[53] Sabine Wilke, Arnab Majumdar, and Washington Y Ochieng. Airport surface operations: A holistic framework for operations modeling and risk management. *Safety Science*, 63:18–33, 2014.

[54] Richard M Zahoransky, Christian Brenig, and Thomas Koslowski. Towards a process-centered resilience framework. In *Availability, Reliability and Security (ARES), 2015 10th International Conference on*, pages 266–273. IEEE, 2015.

[55] WM Zuberek. Timed petri nets definitions, properties, and applications. *Microelectronics Reliability*, 31(4):627–644, 1991.

Published in the Proceedings of the 7th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2017), pp. 128-141, Dec 2017, CEUR vol. 2016 (urn:nbn:de:0074-2016-7)

# Resource Utilization Prediction in Decision-Intensive Business Processes*

Simon Sperl[1], Giray Havur[1,2], Simon Steyskal[1,2], Cristina Cabanillas[2], Axel Polleres[2], and Alois Haselböck[1]

[1] Siemens AG Österreich, Corporate Technology, Vienna, Austria
{name.surname}@siemens.com
[2] Vienna University of Economics and Business, Austria
{name.surname}@wu.ac.at

**Abstract.** An appropriate resource utilization is crucial for organizations in order to avoid, among other things, unnecessary costs (e.g. when resources are under-utilized) and too long execution times (e.g. due to excessive workloads, i.e. resource over-utilization). However, traditional process control and risk measurement approaches do not address resource utilization in processes. We studied an often-encountered industry case for providing large-scale technical infrastructure which requires rigorous testing for the systems deployed and identified the need of projecting resource utilization as a means for measuring the risk of resource under- and over-utilization. Consequently, this paper presents a novel predictive model for resource utilization in decision-intensive processes, present in many domains. In particular, we predict the utilization of resources for a desired period of time given a decision-intensive business process that may include nested loops, and historical data (i.e. order and duration of past activity executions, resource profiles and their experience etc.). We have applied our method using a real business process with multiple instances and presented the outcome.

**Keywords:** decision-intensive business processes, prediction model, resource utilization, risk management

## 1 Introduction

Human resource utilization in an organization can be seen as the proportion of time a person spends on working on allocated tasks. Poor utilization of human resources[3] relates to having resources unnecessarily idle or overloaded. This has a very negative effect on process performance measures such as process completion time, execution costs, and quality [1]. Specifically, while under-utilization of resources leads to higher process execution costs, over-utilization of resources may result in process delays. Therefore, decision makers (typically process managers) should be informed about the utilization

* This work has been funded by the Austrian Research Promotion Agency (FFG) under the project grant 845638 (SHAPE) and the Austrian Science Fund (FWF) under the project grant V 569-N31 (PRAIS).

[3] From now on *resources* for the sake of brevity.

of resources in their organizations for enabling appropriate controls that ensure a desired level of resource utilization.

In a scenario where the process model has no decision nodes, given a baseline schedule and resource allocation [2], deriving the utilization of resources would be trivial. However, actual processes usually have decision points that split the execution flow into different paths so that several cases are projected in the same process model [3]. Decision-intensive processes may contain (nested) loops [4], which actually makes both scheduling and resource allocation more difficult due to the increasing uncertainty. Nonetheless, these kind of processes are common in many domains, e.g. engineering, healthcare, insurance handling, and construction. An inadequate scheduling or allocation of resources may result in a poor resource utilization. While recent resource allocation approaches have already tried and addressed that kind of processes [2], to the best of our knowledge there is a lack of support for resource utilization prediction in such complex scenarios, as most of the existing techniques tend to simplify the application scope [5, 6, 7, 8]. This, in turn, negatively affects risk management in organizations, since process managers miss input that helps to improve the process models and hence, the execution performance of the processes.

In this paper we address that problem and describe a mathematical method for quantifying resource utilization with respect to the structural properties of non-deterministic processes and the historical executions of these proceses. The input values that are used for our prediction model are intrinsically of a stochastic nature, i.e. in the form of probability density functions (PDF). They include, among others, the activity duration PDFs and the resource utilization PDFs. We propagate these input values towards the accumulated utilization function. This function provides a visual overview on the level of future resource utilizations. Therefore, an upcoming over- or under-utilization of resources can be observed. Moreover, we have defined two metrics which characterizes resource over-utilization and under-utilization. We have implemented our approach and demonstrated it with a real process related to large-scale technical infrastructure development and deployment.

We believe that our approach enhances the assessment of process behaviour with the resource perspective. It is especially useful for organizations who need to evaluate the utilization of resources for which they are accountable. With the help of our approach they can automatically get an answer to questions such as whether they have enough resources for the robust execution of their processes, in which periods of time they should expect a delay in the process execution, and when they can safely grant vacations to particular resources, among others.

The paper is structured as follows: Section 2 presents a scenario that motivates this work as well as related work. Section 3 formally defines the input required for our approach. Section 4 describes our approach for predicting resource utilization in decision-intensive business processes. Section 5 applies our method to a real process and presents the outcome, and Section 6 concludes the paper and outlines the future work.

## 2 Background

In the following, we describe an example scenario that motivates this work and shows the problems to be addressed, and then we outline related work.

### 2.1 Running Example

A company that provides large-scale technical infrastructure requires rigorous testing for the systems deployed. Each system consists of different types and number components that are developed and tested in parallel. In order to provide concise and clear examples while describing our method, we use the simple example process shown in Fig. 1. In this process, the *Develop* activity is followed by a *Test* activity, and this may repeat if the test fails. There is also the activity *Manage* which abstracts the potential contractual work running in parallel. There are two resources, namely *Jack* and *Jill*, who execute this process. A more complex process from a real scenario is used later in Section 5 for demostrating the applicability of our method.

### 2.2 Related Work

The approach presented in this paper is mostly related to resource-related risk monitoring and prediction. This risk occurs "due to the high variability that may affect operational processes in real world scenarios" [9]. The risk of inappropriate resource utilization has been identified in [10, 11]. Rosemann et. al. [10] provides a risk taxonomy from the project management perspective. They identify the organizational risk in their taxonomy (e.g., when a resource does not possess the required skills to carry out an activity, or when there is not enough resources to carry out activities on time). Similarly, the process related risk taxonomy of zur Muehlen et. al. [11] contains the resource perspective catagory about lack of resources and/or their skills for activity executions.

Information systems support processes by recording information about their executions in event logs [12]. In order to manage time and resource related risks in an *informed* fashion, a variety of event log mining and prediction mechanisms have been devised: Among others, process duration estimation [13, 14], deadline violation detection [6], resource profiling [7], resource behaviour measurement [5, 15], resource scheduling [16], resource recommendation [17], work prioritization [18], and process performance forecasting [19]. There are also several risk monitoring approaches [8**?** ] combines the scope of several aformentioned mechanisms.

Our method requires extracting durations of activities, and experience of resources for each activity from the event logs for providing realistic resource utilization predictions. Durations can be obtained in a similar way to described by van der Aalst et al. [13] who provide reliable predictions. On the other hand, extraction of resource profiles including their experiences are delineated by Pika et. al. [7].

Within the context of quantifying the resource perspective, our method draws parallels with [15]. We further elaborate our mathematical model for refining our results over the structural properties of running processes. Rather than providing an overall view, Conforti et. al. [8] allow users to take resource-informed decisions at run-time. Our approach is similar to this work in the sense that it reports on resource utilization abnormalities thay may become a problem during the executions of processes.

Folino et al. [19] introduce a performance model for run-time process executions with respect to process variants such as workload and seasonality. Our prediction method can support such models for enriching the context from resource utilization point-of-view.

## 3 Input Data

The following input is required for resource utilization prediction in decision-intensive business processes:

**Input 1** (**Business Process Model**). A business process model $P$ is represented as a directed, connected graph $(N, E)$ with $N = A \cup G \cup \{n_{start}, n_{end}\}$ denoting a finite set of nodes consisting of activities $A$, gateways $G$, and two respective start and end events $n_{start}$ and $n_{end}$, and $E \subseteq N \times N$ representing a set of edges connecting the nodes.

We assume that our model consists of activities, XOR-gateways (decision points), AND-gateways (parallel execution gateways), a start-event, and an end-event. The process always terminates (i.e., it contains no livelocks or deadlocks).

In a real life application, an input process can be composed of the processes that are planned to be executed in the future, and of the unexecuted fragments of the already executing processes.

**Input 2** (**Edge Execution Probability**). Given a process model $P = (A \cup G \cup \{n_{start}, n_{end}\}, E)$, each edge $e \in E$ leaving a XOR-gateway $g \in G_{XOR} \subseteq G$ of $P$ is annotated with an edge execution probability $p_e \in [0, 1]$. Additionally edge execution probabilities $p_e$ must satisfy $\sum_{e \in E \cap (g \times N)} p_e = 1, \ \forall g \in G_{XOR}$.

The outgoing edges of XOR-gateways are annotated with edge execution probabilities (see Fig. 6).

**Input 3** (**Activity Duration PDF**). For each activity $a \in A$, $D_a : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ denotes the PDF representing the duration PDF of activity $a$.

**Example 1.** For clarification, we provide the process in Fig. 1. A single loop of two consequent activities *Develop* and *Test* is presented in parallel with the activity *Manage*. The duration PDFs for *Manage* and *Develop* are normally distributed while *Test* has a fixed duration represented as a solid dot (i.e., a time shifted dirac delta function [4]). The duration of *Test* is *deterministically* 3 time-units (TU) whereas the duration of *Manage* is *on average* 2 TU.

**Input 4** (**Resource Utilization PDF**). Given a set of activities $A$ of a business process $B$ and a set of resources $R$, the resource utilization PDF is $U_{a,r} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ for activity $a \in A$ and resource $r \in R$. $U$ defines the PDFs of the probable additional utilization the execution of an activity causes for all resources.

[4] The Dirac delta function $\delta(x)$ is $\infty$ at $x = 0$ otherwise 0 and satisfies $\int_{-\infty}^{\infty} \delta(x)\,\mathrm{d}x = 1$.

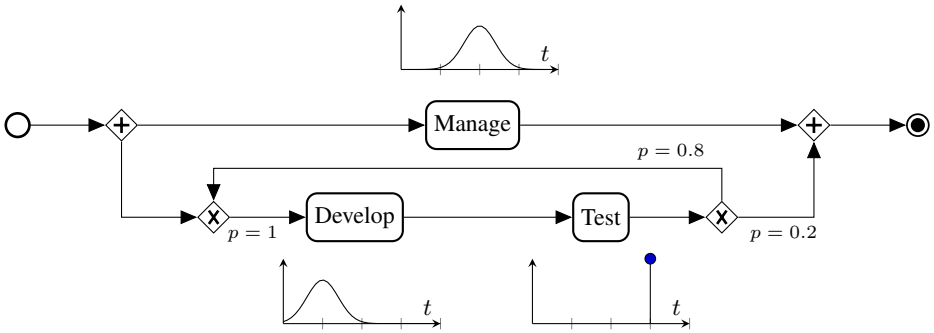



Fig. 1: The running example process with activity duration PDFs $D_a$

| Activities | Resources: Jack | Resources: Jill | Description |
|---|---|---|---|
| *Manage* |  |  | The activity *Manage* utilizes Jack *about* 30%. |
| *Develop* |  |  | The activity *Develop* utilizes Jill *about* 100%, however there is also less of a chance that the same activity may utilize her *about* 50%. |
| *Test* |  |  | The activity *Test* utilizes Jack *about* 85% and Jill *about* 50%. |

Fig. 2: Resource Utilization PDFs for the activities in Fig. 1

The resource utilization PDFs describe which resources are utilized to what extent while executing an activity. Intuitively, one can think of $U_{a,r}(x)dx$ as being the probability of $r$'s utilization falling in the infinitesimal interval $[x, x + dx]$. We assume that utilization values are normalized so that $U_{a,r}(0)$ represents the probability of resource $r$ being $0\%$ utilized by the activity $a$. $U_{a,r}(1)$ specifies the probability of resource $r$'s being utilized $100\%$ by the activity $a$. The notion of utilization can be considered as "the percentage of the work day spent on a task" in our running example.

**Example 2.** See Fig. 2 for an example visualizing the definition of resource utilization PDF. For instance, $U_{Develop,Jill}(0.5) = 0.3$ means with 0.3 probability *Develop* utilizes Jill 50%, and $U_{Develop,Jack}(0) = \infty$ means that Jack is never occupied by *Develop*.

**Input 5 (Experience Matrix).** Given a set of activities $A$ of a process model, and a set of resources $R$, the experience value $X_{a,r} : \mathbb{R}_{>0}$ where $a \in A$ and $r \in R$ is a multiplication factor (a scalar) for activity durations.

The *experience* of each resource $r$ in every activity $a$ is reflected in this matrix. This value theoretically has a range between zero to infinity which is extracted from activity execution durations of resources. Resources that execute activities faster than average have an experience value greater than 1.0.

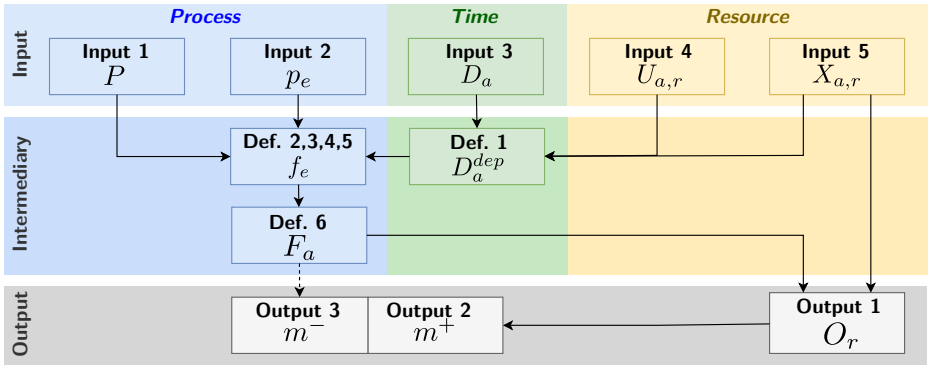



Fig. 3: Input, intermediary functions, and output of our method

We assume that the edge execution probabilities $p_e$, activity duration PDFs $D_a$, resource utilization PDFs $U_{a,r}$, and experience matrix $X$ (*Input 2–5*) are extracted from the event logs obtained from the past executions of the process $P$ (Input 1), where the resources and the durations of the past activity executions are recorded. A solution to this prediction problem is *total utilization PDFs over time* for each resource. Each function provides a utilization prediction for its respective resource.

## 4 Method

Following the problem definition, we first introduce intermediary functions that would allow us to compute the ultimate total utilization PDF, and afterwards we introduce quality metrics that would quantify the risk of abnormal resource utilization that may occur in the future. Fig. 3 is an overview of "*used by*" relation between input values and defined functions of our method (e.g., *Input 3* is used by *Definition 1*). Background colors blue, green and yellow indicates process-related, time-related and resource-related elements.

**Definition 1 (Dependent Activity Duration PDF).** *Given an independent activity duration PDF $D_a$, allocation PDF $U_{a,r}$, experience values $X_{a,r}$ for activities $a$ and resources $r$, the (utilization-and-experience) dependent activity duration PDF $D_a^{dep} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is defined as follows:*

$$x_a = \sum_{r \in R} X_{a,r} \int_0^\infty u U_{a,r}(u) du$$

$$D_a^{dep}(t) = \frac{D_a(t x_a)}{\int_0^\infty D_a(t' x_a) dt'}$$

For each activity, an *experience* value $x_a$ is derived from the experiences of all the resources that are potentially participants in $a$ (i.e., $\exists x \in \mathbb{R}_{>0} : U_{a,r}(x) > 0$). $x_a$ is

used as a division factor for $D_a$, therefore the experience values above 1.0 reduce the width of $D_a$ (i.e., they act as speed-up factor for execution times), and the opposite holds for the values smaller than 1.0.

### 4.1 Edge Transition Probability Function

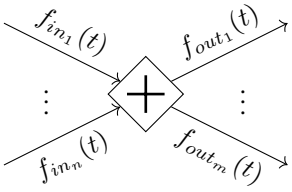
In order to compute how often an activity is being executed, we need to extract the probability values of edge traversals over time. Only activities can create time delays via their duration PDFs. Edge transitions are always instantaneous.

**Definition 2 (Edge transition probability function).** *$f_e(t) : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ denotes the probability that an edge $e \in E$ is traversed at time $t \in \mathbb{R}_{\geq 0}$.*

$$
\begin{aligned}
&P(\textit{An edge is traversed between } t_1 \textit{ and } t_2) \\
&= P(\textit{Edge is traversed before } t_2) - P(\textit{Edge is traversed before } t_1) \\
&= \int_0^{t_2} f(t')\,dt' - \int_0^{t_1} f(t')\,dt'
\end{aligned}
$$

Note that in general $f_e$ is not a PDF. However, if the process contains no XOR-gateways all $f_e$ are PDFs since $\forall e \in E : \int f_e(t)\,d\mathrm{t} = 1$.

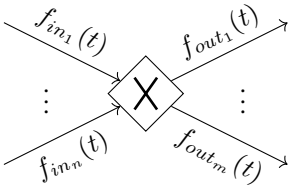
**Definition 3 ($f$ for AND-Gateway).** *Given $g \in G_{and} \subseteq G$ with incoming edges $\{in_1, \ldots, in_n\} = E \cap (N \times g)$ and outgoing edges $\{out_1, \ldots, out_m\} = E \cap (g \times N)$, the edge transition probability function $f_e(t)$ for outgoing edges $e$ is defined as*




$$f_{out_j}(t) = \left( \prod_{i=1}^{n} \int^{t} f_{in_i}(t')\,dt' \right) \frac{d}{dt} \quad , 1 \leq j \leq m$$

Having a generic analytical result about the edge transition behavior of the process is difficult unless the input is in form of time-shifted dirac delta functions or PDFs of an exponential distribution.

**Definition 4 ($f$ for XOR-Gateway).** *Given $g \in G_{XOR} \subseteq G$ with incoming edges $\{in_1, \ldots, in_n\} = E \cap (N \times g)$, outgoing edges $\{out_1, \ldots, out_m\} = E \cap (g \times N)$, and edge execution probabilities $\{p_{out_1}, \ldots, p_{out_m}\}$, the edge transition probability function $f_e(t)$ for outgoing edges $e$ is defined as*




$$f_{out_j}(t) = p_{out_j} \sum_{i=1}^{n} f_{in_i}(t) \quad , 1 \leq j \leq m$$

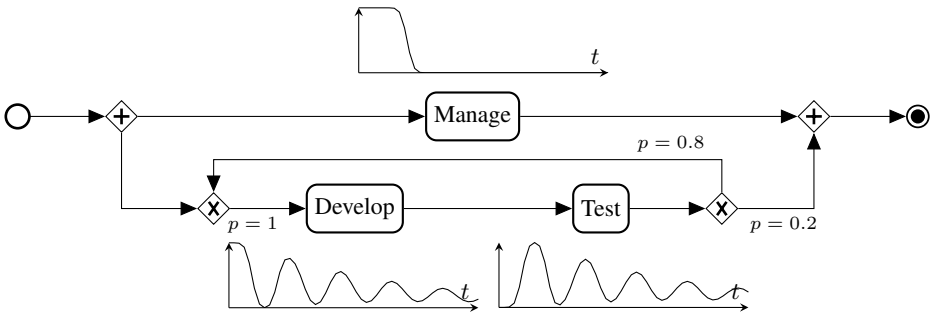
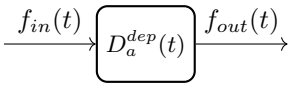



Fig. 4: The running example process with activity execution prob. functions $F_a$

**Definition 5 ($f$ for Activities).** *Given an activity $a \in A$ with one incoming edge $e_{in}$, one outgoing edge $e_{out}$, and activity duration PDF $D_a^{dep}$, the edge transition probability function $f_{out}(t)$ is defined as*




$$f_{out}(t) = f_{in}(t) * D_a^{dep}(t)$$

Note that the convolution operator $*$ for two functions $f$ and $g$ is defined as $(f * g)(t) = \int f(t')g(t - t')\,\mathrm{d}t'$.

We compute edge transition probability functions directly on the process. Another way of doing this is described in [20]. Their method requires decomposition of the process into process blocks.

### 4.2 Estimating Total Resource Utilization

In order to be able to define the total resource utilization function, we need to know the probability density of an activity $a$ being executed at time $t$. From edge transition probabilities for the incoming edge of $a$, the probability density that an activity is executed for each point in time is represented by the activity execution probability function $F_a(t)$.

**Definition 6 (Activity execution probability function).** *The function $F_a(t) : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ denotes the probability density that an activity $a \in A$ is executed at time $t \in \mathbb{R}_{\geq 0}$.*

$$\begin{aligned} F_a(t) &= P(\textit{Activity is currently being executed at } t) \\ &= P(\textit{Activity is entered before } t) - P(\textit{Activity is left before } t) \\ &= \int_0^t f_{in}(t')\,dt' - \int_0^t f_{in}(t') * D_a^{dep}(t')\,dt' = \int_0^t f_{in}(t')\,dt' - \int_0^t f_{out}(t')\,dt' \end{aligned}$$

**Example 3.** Our running example with the activity duration PDFs $D_a$ in Fig. 1 would then result into the activity execution probability functions $F_a$ presented in Fig. 4. *Manage* is immediately executed once. As *Develop* and *Test* repeat (infinitely) in the loop, each repetition lowers the execution probability of the next iteration.

**Output 1** (Total utilization PDF over time)**.** $O_r$ is the utilization PDF of a resource $r \in R$ at time $t \in \mathbb{R}_{\geq 0}$, where $A = \{a_1, \ldots, a_n\}$. A value $O_r(t, u)$ represents the probability density that $r$ is utilized by an amount of $u$ percent of his/her time (e.g. $u = 1$ means full time, $u > 1$ means over-utilization) at time $t$. The operator $*_u$ is the convolution over the parameter $u$.

$$
\begin{aligned}
O_r &: (\mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0}) \to \mathbb{R}_{\geq 0} \\
O_r(t, u) &= \delta(u) \\
&\quad *_u \ (U_{a_1,r}(u) F_{a_1}(t) + \delta(u)(1 - F_{a_1}(t))) \\
&\quad \ldots \\
&\quad *_u \ (U_{a_n,r}(u) F_{a_n}(t) + \delta(u)(1 - F_{a_n}(t)))
\end{aligned}
$$

For each resource, there is one total utilization PDF over time. In these PDFs, the activity execution probability functions and resource utilization PDFs of the respective resources are combined.

**Example 4.** In Fig. 5, the total utilization PDF over time for Jack $O_{Jack}$ is based on (1) our running example with the activity execution probability functions $F$ as seen in Fig. 4, and (ii) the resource utilization PDFs $U$ as seen in Fig. 2. $O_{Jack,Test}(t, u) = (W_{Jack,Test}(u) F_{Test}(t) + \delta(u)(1 - F_{Test}(t)))$ is the probable utilization of Jack by the activity *Test* shown in top-left corner. In a similar way, $O_{Jack,Manage}$ is on the top-right corner. We do not show $O_{Jack,Develop}$, since it has no influence on Jack's total utilization (see Fig. 2). We can clearly observe in $O_{Jack,Manage}$ that the activity *Manage* is non-repeating. The combined total utilization PDF for Jack $O_{Jack}$ is shown in bottom of the Figure. It shows how the probable utilizations of Jack's activities combine into a period of over-utilization as shown in the red region. Such region representing over-utilization are of special interest for the decision makers (e.g., project managers) who are also responsible for time and resource management.

### 4.3 Quality Metrics

Based on resource utilization PDFs $O_r$, we can define various metrics for quantifying abnormal resource utilization. These metrics can be used as optimization criteria for managing organizations and their schedules.

**Output 2** (Resource over-utilization metric)**.** The resource over-utilization metric ${m^+}_r : \mathbb{R}_{\geq 0}$ for a resource $r \in R$ is the volume of utilization larger than 1.

$$
{m^+}_r = \int_0^\infty \int_1^\infty o O_r(t, u) \, \mathrm{d}u \, \mathrm{d}t
$$

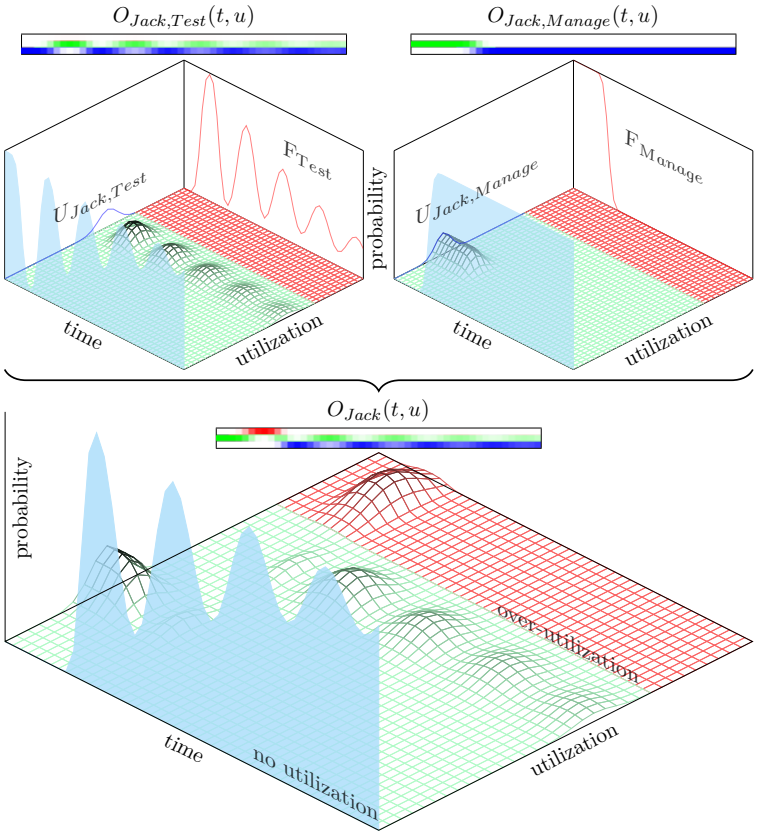



Fig. 5: Total Utilization PDF over time $O_{\texttt{Jack}}$ of our running example and the relevant pieces it is composed of. Directly above the plots, we provide a more condensed visualization where the x-axis is time, and the y-axis is utilization. The color intensity is directly proportional with the probability density value. Note that the blue parts are of infinite slope caused by $\delta(u)$.

The total over-utilization $m$ of a process is the sum of over-utilization of all resources: $m^+ = \sum_{r \in R} {m^+}_r$

In other words, $m^+$ is a value characterizing resource over-utilization for a whole organization (i.e., for all resources and the processes involved).

**Output 3** (Resource under-utilization metric)**.** The resource under-utilization metric ${m^-}_r$ for a resource $r \in R$ is the accumulated volume under the optimal resource occu-

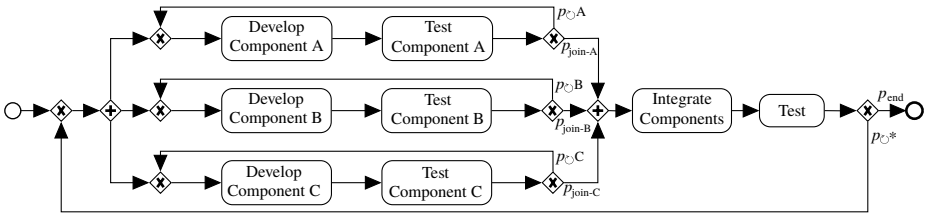



Fig. 6: An often-encountered development process with multiple parallel "Develop and Test" cycles with a final integration step

pancy threshold on the occupancy axis (cf. 1.0).

$$F_P(t) = \int_0^t f_{e_{start}}(t')\,\mathrm{d}t' - \int_0^t f_{e_{end}}(t')\,\mathrm{d}t'$$

$$m^-{}_r = \int_0^\infty F_P(t) \int_0^1 |1-u| O_r(t,u)\,\mathrm{d}u\,\mathrm{d}t$$

In other words, $m^-$ is a value characterizing resource under-utilization for a whole organization (i.e., for all resources and the processes involved). $F_P$ in the formula above is similar to $F_a$ in Definition 6 in the sense that $F_P$ denotes the probability that a process $P$ is executed at time $t \in \mathbb{R}_{\geq 0}$.

## 5 Application to a Real Process

We tested our method in a setting where the utilization of 10 resources is forecasted in 10 process instances of the process shown in Fig. 6. In this realistic process, each system consists of different types and number of components that are developed and tested in parallel. All *Develop* activities must be tested independently, and repeated development effort is expected. Additionally, a final *Integrate* and *Test* phase is mandatory for the combined system, which may cause additional repetitions of the entire process. It is expected for resources to work on multiple processes in parallel.

Table 1 describes the properties of 10 process instances with different starting times and resource utilizations: The process name (**id**), the starting time ($t_s$), the initial letter of the resource allocated to a certain activity ($r$), mean duration of the activity ($\partial$), repetition probability of the iteration "*Develop* and *Test*" for component $X$ ($p_{\circlearrowright X}$) are given (cf. $p_{\circlearrowright *}$ is the edge execution probability of restarting the process). The different components of each process have different failure rates and duration distributions.

In order to generate the utilization functions for each resource, we simulate the process instances with respect to their properties (cf. Table 1) and apply the methods described previously. Note that we leave out the process execution paths with a probability lower than 0.1%.

Fig. 7 shows the visualization of the utilization functions for each resource. For instance, Boris is overoccupied most of the times because he is allocated to 5 Develop, 3 Test, and 4 Integration activities which are overlapping. Grace becomes overoccupied

Table 1: Properties of process instances

| | | $Dev_A$ | | $Test_A$ | | | $Dev_B$ | | $Test_B$ | | | $Dev_C$ | | $Test_C$ | | | $Int_*$ | | $Test_*$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **id** | $t_s$ | $r$ | $\partial$ | $r$ | $\partial$ | $p_{\circlearrowright A}$ | $r$ | $\partial$ | $r$ | $\partial$ | $p_{\circlearrowright B}$ | $r$ | $\partial$ | $r$ | $\partial$ | $p_{\circlearrowright C}$ | $r$ | $\partial$ | $r$ | $\partial$ | $p_{\circlearrowright *}$ |
| 1 | 40 | B | 50 | G | 20 | 0.6 | E | 12.5 | H | 5 | 0.3 | I | 25 | J | 10 | 0.3 | J | 22.5 | C | 9 | 0.1 |
| 2 | 40 | H | 40 | G | 16 | 0.3 | I | 10 | H | 4 | 0.6 | E | 20 | H | 8 | 0.1 | F | 18 | I | 7.2 | 0.15 |
| 3 | 40 | F | 70 | E | 28 | 0.2 | C | 17.5 | J | 7 | 0.6 | C | 35 | F | 14 | 0.2 | B | 31.5 | I | 12.6 | 0.1 |
| 4 | 40 | J | 60 | I | 24 | 0.2 | G | 15 | H | 6 | 0.5 | E | 30 | F | 12 | 0.3 | B | 27 | I | 10.8 | 0.1 |
| 5 | 40 | B | 40 | C | 16 | 0.6 | B | 10 | B | 4 | 0.3 | I | 20 | B | 8 | 0.3 | B | 18 | I | 7.2 | 0.1 |
| 6 | 70 | G | 20 | A | 8 | 0.4 | E | 5 | F | 2 | 0.4 | E | 10 | F | 4 | 0.4 | B | 9 | G | 3.6 | 0.2 |
| 7 | 100 | B | 80 | E | 32 | 0.4 | E | 20 | H | 8 | 0.8 | C | 40 | B | 16 | 0.1 | D | 36 | A | 14.4 | 0.1 |
| 8 | 110 | H | 40 | I | 16 | 0.5 | I | 10 | J | 4 | 0.8 | A | 20 | D | 8 | 0.5 | D | 18 | G | 7.2 | 0.1 |
| 9 | 120 | H | 60 | C | 24 | 0.6 | G | 15 | J | 6 | 0.3 | A | 30 | H | 12 | 0.5 | J | 27 | C | 10.8 | 0.1 |
| 10 | 170 | B | 30 | G | 12 | 0.7 | E | 7.5 | F | 3 | 0.6 | E | 15 | D | 6 | 0.5 | D | 13.5 | E | 5.4 | 0.3 |

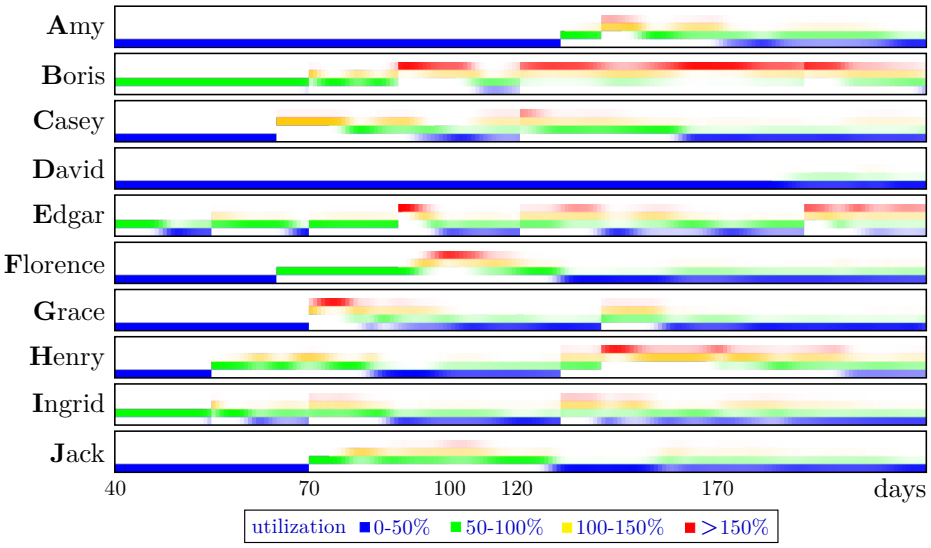



Fig. 7: Compact visualization of resource utilization forecast with 10 resources and 10 projects of type Figure 6 over one year

due to the start of project-6 at day 70. Edgar, Florence, and Henry are also expected to be visibly overoccupied in some periods during the execution of these 10 projects, though their situations are not as critical as Boris's, because their activities are not as many times overlapping as the activities of Boris. David's utilization looks exceptionally low. Moreover, by using the quality metrics defined in Section 4.3, we are informed that there is a 10 times greater risk of resource under-occupancy ($m^- = 2665.45$) than resource over-occupancy ($m^+ = 273.44$). This can be intuitively confirmed by comparing the amount of blue and red areas on Fig. 7.

Fig. 7 and the quality metrics suggest that: (i) There are more resources than needed in this setting, and (ii) demand for resources can still be balanced, especially those of Boris's for a more robust execution.

The computational complexity of our method is equivalent to the complexity of numerical integration which is P-complete [21]. Its implementations perform in quasi-

linear time. Therefore, our method responds to the problems with real-world sizes in a few seconds, and it is suitable for both design-time and run-time prediction tasks.

## 6 Conclusions and Future Work

In this paper we have introduced a novel method for predicting resource utilization in decision-intensive business processes. Our approach facilitates to obtain a resource utilization overview for the whole organization where the processes have a stochastic nature. As a result, the decision makers are provided with actual insights about their organizational feasibility. One advantage of our approach is that it can be readily used in practice, and it can be incorporated in BPM systems as a supplementary risk monitoring element.

Our future work primarily involves conducting exhaustive evaluations to assess the applicability of the approach in more real settings and compare the performance results. We also aim at integrating the approach into a BPM system as well as at adapting the current design-time method to be used at run time too, i.e. to make it more dynamic such that the resource utilization predictions are updated during the execution of the process instances.

## References


[1] Michael Rosemann and Jan vom Brocke. The six core elements of business process management. In *Handbook on business process management 1*, pages 105–122. Springer, 2015.

[2] Giray Havur, Cristina Cabanillas, Jan Mendling, and Axel Polleres. Resource Allocation with Dependencies in Business Process Management Systems. In *BPM (Forum)*, volume 260, pages 3–19, 2016.

[3] Marlon Dumas, Marcello La Rosa, Jan Mendling, Hajo A Reijers, et al. *Fundamentals of Business Process Management*, volume 1. Springer, 2013.

[4] Qiang Liu and Jin He. Study of complex loop patterns based on time petri net. In *Computer Supported Cooperative Work in Design, 2007. CSCWD 2007. 11th International Conference on*, pages 801–805. IEEE, 2007.

[5] Zhengxing Huang, Xudong Lu, and Huilong Duan. Resource behavior measure and application in business process management. *Expert Systems with Applications*, 39(7):6458–6468, 2012.

[6] Anastasiia Pika, Wil MP van der Aalst, Colin J Fidge, Arthur HM ter Hofstede, and Moe T Wynn. Predicting deadline transgressions using event logs. *Lecture Notes in Business Information Processing*, 132:211–216, 2012.

[7] Anastasiia Pika, Michael Leyer, Moe T Wynn, Colin J Fidge, Arthur HM Ter Hofstede, and Wil MP Van Der Aalst. Mining resource profiles from event logs. *ACM Transactions on Management Information Systems (TMIS)*, 8(1):1, 2017.

[8] Raffaele Conforti, Massimiliano de Leoni, Marcello La Rosa, Wil MP van der Aalst, and Arthur HM ter Hofstede. A recommendation system for predicting

risks across multiple business process instances. *Decision Support Systems*, 69:1–19, 2015.

[9] Paolo Ceravolo, Antonia Azzini, Ernesto Damiani, Mariangela Lazoi, Manuela Marra, and Angelo Corallo. Translating process mining results into intelligible business information. In *Proceedings of the The 11th International Knowledge Management in Organizations Conference on The changing face of Knowledge Management Impacting Society*, page 14. ACM, 2016.

[10] Michael Rosemann and Michael Zur Muehlen. Integrating risks in business process models. *ACIS 2005 Proceedings*, page 50, 2005.

[11] Michael Zur Muehlen and Danny Ting-Yi Ho. Risk management in the bpm lifecycle. In *International Conference on Business Process Management*, pages 454–466. Springer, 2005.

[12] Wil MP van der Aalst. *Process mining: data science in action*. Springer, 2016.

[13] Wil MP Van der Aalst, M Helen Schonenberg, and Minseok Song. Time prediction based on process mining. *Information systems*, 36(2):450–475, 2011.

[14] Andreas Rogge-Solti and Mathias Weske. Prediction of business process durations using non-markovian stochastic petri nets. *Information Systems*, 54:1–14, 2015.

[15] Marijke Swennen, Niels Martin, Gert Janssenswillen, Mieke J Jans, Benoît Depaire, An Caris, and Koen Vanhoof. Capturing resource behaviour from event logs. In *SIMPDA*, 2016.

[16] Arik Senderovich, Matthias Weidlich, Avigdor Gal, and Avishai Mandelbaum. Mining resource scheduling protocols. In *International Conference on Business Process Management*, pages 200–216. Springer, 2014.

[17] Michael Arias, Eric Rojas, Jorge Munoz-Gama, and Marcos Sepúlveda. A framework for recommending resource allocation based on process mining. In *International Conference on Business Process Management*, pages 458–470. Springer, 2015.

[18] Suriadi Suriadi, Moe T Wynn, Jingxin Xu, Wil MP van der Aalst, and Arthur HM ter Hofstede. Discovering work prioritisation patterns from event logs. *Decision Support Systems*, 2017.

[19] Francesco Folino, Massimo Guarascio, and Luigi Pontieri. Discovering context-aware models for predicting business process performances. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 287–304. Springer, 2012.

[20] Wil MP Van der Aalst, Kees M Van Hee, and Hajo A Reijers. Analysis of discrete-time stochastic petri nets. *Statistica Neerlandica*, 54(2):237–255, 2000.

[21] Akitoshi Kawamura. Computational complexity in analysis and geometry. *University of Toronto, Toronto, Ont., Canada*, 2011.

Published in the Proceedings of the 27th International Conference on Cooperative Information Systems (CoopIS 2019), pp. 533-551, Oct 2019, Springer LNCS vol. 11877

# History-Aware Dynamic Process Fragmentation for Risk-Aware Resource Allocation*

Giray Havur[1,2] and Cristina Cabanillas[1]

[1]Vienna University of Economics and Business, Austria
{name.surname}@wu.ac.at
[2]Siemens AG Österreich, Corporate Technology, Vienna, Austria
{name.surname}@siemens.com

**Abstract.** Most Process-Aware Information Systems (PAIS) and resource allocation approaches do the selection of the resource to be allocated to a certain process activity at run time, when the activity must be executed. This results in cumulative (activity per activity) local optimal allocations for which assumptions (e.g. on loop repetitions) are not needed beforehand, but which altogether might incur in an increase of cycle time and/or cost. Global optimal allocation approaches take all the process-, organization- and time-related constraints into account at once before process execution, handling better the optimization objectives. However, a number of assumptions must be made upfront on the decisions made at run time. When an assumption does not hold at run time, a resource reallocation must be triggered. Aiming at achieving a compromise between the pros and cons of these two methods, in this paper we introduce a novel approach that fragments the process dynamically for the purpose of risk-aware resource allocation. Given historical execution data and a process fragmentation threshold, our method enhances the feasibility of the resource allocations by dynamically generating the process fragments (i.e. execution horizons) that satisfy the given probabilistic threshold. Our evaluation with simulations demonstrates the advantages in terms of reduction in reallocation efforts.

**Keywords:** Business process management, dynamic process fragmentation, quasi-online scheduling, resource allocation, process mining

## 1 Introduction

Resource allocation in business processes selects the resource(s) responsible for a process activity at run time among the suitable resources according to predefined assignment criteria (e.g. based on roles or other organizational properties)[1]. Despite being a necessary functionality in any Process-Aware Information System (PAIS), current systems provide limited support for resource allocation. Generally, actual resource allocation is delegated to people to some extent, in the best case by implementing Push/Pull

* Funded by the Austrian Science Fund (FWF) Elise Richter programme under agreement V 569-N31 (PRAIS).

[1] In this work we assume human resources but the concept could be adapted for non-human resource allocation, too.

patterns [17] (e.g. by offering the activity to all the suitable resources and letting them decide who will become responsible for its execution – distribution by offer). This behavior tends to overlook optimization functions regarding performance metrics such as process cycle time or execution cost. On the other hand, when it is up to the process manager to decide who to allocate to the next activities, they get no information about the horizon until which the allocations will hold. When decision points are present in the process, assumptions on the respective choices must be made beforehand. The bigger the uncertainty in the process (i.e. the number of decisions), the more critical this issue becomes. While taking a higher risk (i.e. allocating more activities at once) may be beneficial as for process cycle time or cost, if the assumptions made do not hold at run time the opposite effects may appear and *undesired* reallocations may be necessary. On the contrary, a very conservative (less risky) approach that allocates resources only when it is required (i.e. before activity execution) avoids speculating about potential future behaviors of the process but does not assure the most optimal (or even feasible) results at the process level and can incur in non-anticipated delays or deadlocks. For instance, if there is a binding of duties between $A$ (executed first) and $B$, and at the moment of allocating $B$ the required resource is not available, the process will be delayed.

In order to avoid manual resource allocation, several approaches propose more advanced and automated techniques [21, 16, 19, 4]. Some find out the most adequate resource at the moment of executing an activity, hence allocating the process *activity per activity* at run time to the "best" resource available [21, 16] (also known as *online scheduling* [18]). Others aim for a global optimal solution [19, 4] where the entire process is considered for allocation before it is started (also known as *offline scheduling* [18]), so the resources for all the activity instances are selected at once on the basis of the present process-, organization- and time-related constraints. In terms of risk management, these techniques lie in the two extremes of the spectrum, and to the best of our knowledge there is not yet an approach handling risk and uncertainty in a more flexible way in this context.

In this paper we introduce a novel approach to dynamically create and allocate *process fragments* at run time being aware of the risks taken when making resource allocation decisions. The novelties of the approach include: (i) the use of historical execution data for making probabilistic decisions ahead of time; and (ii) the dynamic fragmentation of the process based on a fragmentation threshold that provides flexibility in terms of risk management and helps to prevent potential reallocations. We have implemented the approach using an encoding based on Answer Set Programming (ASP) [3] that proved to produce good performance results in optimal resource allocation [4]. We have evaluated our fragmentation method with simulations aimed to assess its behaviour with respect to reallocation needs. The findings support our hypothesis that the feasibility of the resource allocations improves when the risk is kept low.

The paper is structured as follows. Section 2 describes an example scenario that illustrates the research problem. Section 3 defines concepts needed to understand our approach. Section 4 introduces our new process fragmentation method. Section 5 explains the implementation of the method and the evaluation performed. Section 6 provides a more detailed description of the state-of-the-art. Finally, Section 7 outlines the conclusions drawn from the work along with its limitations and potential future steps.

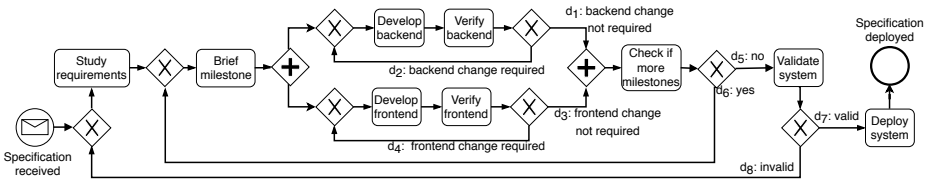



Fig. 1: BPMN model of the software development process

## 2 Motivating Scenario

A manager of a software development company would like to handle the allocation of their (human) resources to every new project. They developed and refined a business process model for the execution of software development projects as illustrated with Business Process Model and Notation (BPMN) [12] in Fig. 1. After a project specification is received, the requirements of the project are studied and a number of milestones are derived. Every milestone involves two separate development parts (frontend and backend) that are concurrently undertaken and which, in turn, may require one or more iterations depending on the respective verification results. Once all the milestones are fully achieved, the developed software needs to be validated against the requirements and specifications received at the beginning of the project. If the validation fails, the study of the requirements must be reinitiated and new milestones are developed.

The manager is aware of the fact that the process has multiple decision points (represented with XOR split gateways in BPMN) where their forecasting on the allocations of resources deviated many times in the past. This forces them to reallocate resources from scratch to the upcoming activities at each deviation. With the support of a PAIS, they keep track of previous executions of the software development process in event logs. This helps them to infer the duration of the activities depending on the resources allocated to them. However, they found the need for an automated mechanism that could perform their manual allocations in a more flexible and less risky way by leveraging the process-, organizational- and temporal-related data available, aiming to achieve time- and cost-effective executions and avoid the reallocations to a bigger extent.

## 3 Preliminaries

BPMN [12] is the de-facto standard process modeling notation due to its understandability and ease of use. However, because of their well-defined semantics and their analysis capabilities, in this work we rely on Petri nets [13] satisfying the workflow properties [22] for process modeling. Nonetheless, note that many process modeling notations, including BPMN, can be automatically mapped to Petri nets [11].

A Petri net is a bipartite graph composed of *places* and *transitions*. The places might contain tokens, whose distribution might change over time.

**Definition 1 (Petri net).** A Petri net is a 4-tuple $PN = (P, T, F, \nu)$, where:

- $P = \{p_1, p_2, ..., p_n\}$ is the set of *places*, represented graphically as circles,

- $T = \{t_1, t_2, ..., t_n\}$ is the set of *transitions*, represented graphically as rectangles,
- $F \subseteq (P \times T) \bigcup (T \times P)$ is the set of *arcs* (flow relations), represented as arrows,
- $\nu : F \rightarrow \mathbb{Z}^+$ is the arc weight mapping indicating cardinality constraints on the movement of tokens throughout the net.

The *input places* and the *output places* of each transition $t \in T$ are ${}^\bullet t = \{p \in P \mid (p,t) \in F\}$ and $t^\bullet = \{p \in P \mid (t,p) \in F\}$, respectively. Similarly, the *input transitions* and the *output transitions* of each place $p \in P$ are ${}^\bullet p = \{t \in T \mid (t,p) \in F\}$ and $p^\bullet = \{t \in T \mid (p,t) \in F\}$, respectively. A *marking* (or *state*) $M = \{\mu(p_1), \mu(p_2), ..., \mu(p_{|P|})\}$, $\mu : P \rightarrow \mathbb{Z}_{\geq 0}$ represents the distribution of tokens over the set of places. When it assigns a non-negative integer *k* to place *p*, we say that *p* is marked with *k* tokens. Pictorially, we place *k* black dots in place *p*. A Petri net with an initial marking, which represents the initial distribution of tokens, is a *Petri net system*.

**Definition 2 (Petri net system).** A Petri net system is a tuple $PNS = (P, T, F, \nu, M_0)$, where $(P, T, F, \nu)$ is a Petri net, and $M_0$ is its *initial marking*.

The initial marking of the Petri net $M_0$ can change into successor markings. These changes are described as *firing rules*. Such rules introduce a dynamic aspect to the Petri net by modifying its state, giving rise to the net behavior. A transition $t$ is *enabled* when there are at least as many tokens as $\nu(p,t)$ in each input place $p \in {}^\bullet t$. An enabled transition can therefore *fire*. The number of tokens that are added to each output place after firing of $t$ is defined as $\nu(t,p)$. The firing of a transition changes the current marking by subtracting $\nu(t,p)$ amount of tokens from each input place $p \in {}^\bullet t$ and adding $\nu(p,t)$ amount of tokens to each output place $p \in t^\bullet$, and hence, it moves the net from a marking $M_{k-1}$ to a new marking $M_k$ denoted as $M_{k-1} \xrightarrow{t_k} M_k$. A *firing sequence* of transitions $\sigma = t_1, t_2, ..., t_n$ changes the state of the Petri net at each firing: $M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} ... \xrightarrow{t_n} M_n$. A marking $M_k$ is *reachable* if there is a sequence $\sigma$ such that $M_0 \xrightarrow{\sigma} M_k$ (i.e. from the initial marking to $M_k$).

Petri nets are classified according to several criteria, including cardinality and behavioral constraints. The Petri nets that represent the business processes addressed in this work have the following properties:

- They constitute a so-called *workflow net* [22], which means that they contain a starting place $p_s$ such that ${}^\bullet p_s = \emptyset$ and an ending place $p_e$ such that $p_e^\bullet = \emptyset$; and they are *connected*, that is, every node in the Petri net is on the path from $p_s$ to $p_e$.
- They are *1-safe*, which means that each place contains at most one token at any state (i.e. for any place $p \in P$, $0 \leq \mu(p) \leq 1$).
- They are *free-choice*, which implies that the choice between multiple transitions can never be influenced by the rest of the net (i.e. for any two different places $\{p_i, p_j\} \subseteq P$, $(p_i^\bullet \cap p_j^\bullet) = \emptyset$ or $p_i^\bullet = p_j^\bullet$),

Besides, the transitions of the Petri net represent process activities (tasks) and the places represent states of the business process. Some especial transitions, called *silent transitions* and colored in black, are incorporated for modeling specific process behavior. Therefore, to give more meaningful labels to the nodes of the Petri net, given the *alphabet* (set of labels) $\Sigma$, the function $\lambda : T \rightarrow \Sigma \cup \{\epsilon\}$ assigns to each transition $t \in T$ either a symbol from $\Sigma$ or the empty string $\epsilon$ (for silent transitions). Fig. 2 depicts the Petri net corresponding to the BPMN model in Fig. 1. For the purpose of this

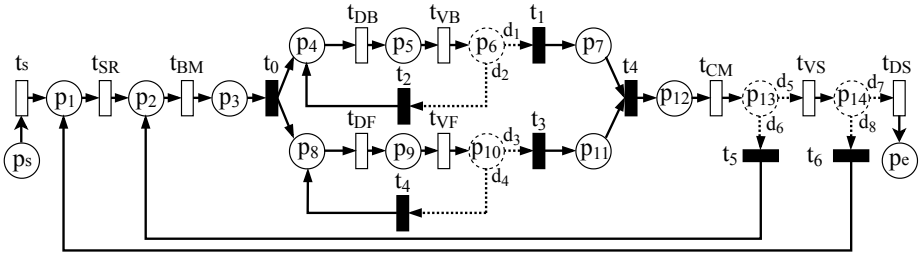



Fig. 2: Petri net model of the software development process

paper, dashed places denote *decision points*. A place $p$ is a decision point if $|p^\bullet| > 1$. At run time, a non-deterministic decision is made on the output transitions $p^\bullet$ of a decision point $p$, and dashed arcs connected to these places denote different arcs that lead to different transitions (i.e. *decisions*) from which only one is to be fired during the execution of the system (e.g. decision $d_1$ in Fig. 1 corresponds to the flow arc $(p_6, t_1)$ in Fig. 2).

Three types of ordering relations can be identified between transitions of a Petri net. Specifically, given two transitions $t_x, t_y \in T$ of a Petri net, then:

- $t_x$ *directly precedes* $t_y$ ($t_x \rightarrow t_y$) if the net contains a path with *exactly* two arcs (i.e. $(t_x, p_i) \cup (p_i, t_y) \subseteq F$) leading from $t_x$ to $t_y$ (e.g. $t_{DB}$ and $t_{VB}$ in Fig. 2). This precedence relation is reduced to cover activity transitions in the *direct activity precedence* relation ($t_x \xrightarrow{a} t_y$), such that $t_j \rightarrow t_{j+1} \rightarrow \ldots \rightarrow t_n$, where only $t_j$ and $t_n$ are activity transitions (e.g. $t_{BM} \xrightarrow{a} t_{DB}$ and $t_{VB} \xrightarrow{a} t_{CM}$ in Fig. 2).
- $t_x$ *is in conflict with* $t_y$ ($t_x \# t_y$) when the Petri net contains a place $p$ where $\{t_x, t_y\} \subseteq p^\bullet$ (e.g. $t_1$ and $t_2$ in Fig. 2). A conflict-free Petri net system has no two transitions in conflict relation.
- $t_x$ *can be executed parallelly with* $t_y$ ($t_x || t_y$) if $t_x$ and $t_y$ are neither in preceding nor in excluding relation (e.g. $t_{DB}$ and $t_{DF}$ in Fig. 2).

The selection of the resources that are suitable to execute a process activity is done by defining assignment constraints based on organizational and process-related information. Different types of organizational structures lead to different organizational models [8]. In this work we assume an organizational structure based on roles following the Role-Based Access Control (RBAC) model [2].

**Definition 3 (RBAC Model).** An RBAC Model is a 6-tuple $O = (A, R, L, S_{AL}, S_{RL}, S_{LL})$, where:

- $A$ is the set of *activities* that corresponds to the activity transitions in a $PNS = (P, T, F, \nu, M_0)$ that represents an executable business process, hence $A \subseteq T$,
- $R$ is the set of *resources*,
- $L$ is the set of *roles*,
- $S_{AL} \subseteq 2^{(A \times L)}$ is the set of activity-to-role assignment tuples specifying which activity can be executed by the resources associated with which role(s),
- $S_{RL} \subseteq 2^{(R \times L)}$ is the set of resource-to-role assignment tuples identifying the roles of a resource,

- $S_{LL} \subseteq 2^{(L \times L)}$ is the set of role-to-role assignment tuples that creates a hierarchical structure. The symbol $\preceq$ indicates the ordering operator. If $l1 \preceq l2$, then $l1$ is referred to as the *senior* of $l2$ and therefore, the resources of $l1$ can also execute the activities assigned to $l2$. Conversely, $l2$ is the *junior* of $l1$. Note that for any $l_i \preceq l_{i+1} \preceq ... \preceq l_j,\ l_n \npreceq l_m$ where $i \leq m \leq n \leq j$.

For example, given the RBAC model in Table 1 for our running example, the activity *Study requirements* must be performed by managers and/or coordinators. Therefore, Amy, Oliver and Emma might be involved in its execution. However, we also need to know how many resources of each type are required.

**Definition 4 (Resource Requirement).** A resource requirement $q \subseteq 2^{(L \times \mathbb{Z}_{\geq 0})}$ is a set of binary relations that represents the number of resources with role $l \in L$ required.

| $(a, l) \in S_{AL}$ | $(r, l) \in S_{RL}$ |
|---|---|
| {Study req., Manager} | {Amy, Manager} |
| {Study req., Coordinator} | {Oliver, Coordinator} |
| {Brief mile., Coordinator} | {Emma, Coordinator} |
| {Dev. backend, Coder} | {Glen, Software eng.} |
| {Dev. backend, Software eng.} | {Evan, Testing exp.} |
| {Verify backend, Testing exp.} | {Mia, Testing exp.} |
| {Verify backend, Software eng.} | {Drew, UI designer} |
| {Dev. frontend, UI designer} | {Ellen, UI designer} |
| {Dev. frontend, Coder} | {Jessie, Coder} |
| {Verify frontend, Testing exp.} | {Liam, Coder} |
| {Verify frontend, Software eng.} | {Alex, Coder} |
| {Check mile., Coordinator} | |
| {Validate syst., Manager} | |
| {Validate syst., Software eng.} | |
| {Deploy syst., Manager} | |

Table 1: RBAC model of the software company

| $a$ | $q_a$ | $\delta(a, q_a)$ |
|---|---|---|
| Study req. | {(Manager,1)} | 20 |
| Study req. | {(Manager,1),(Coordinator,1)} | 15 |
| Brief mile. | {(Coordinator,1)} | 1 |
| Dev. backend | {(Software eng.,1),(Coder,2)} | 10 |
| Verify backend | {(Testing exp.,1),(Software eng.,1)} | 4 |
| Dev. frontend | {(UI designer,1),(Coder,1)} | 12 |
| Dev. frontend | {(UI designer,1),(Coder,2)} | 8 |
| Verify frontend | {(Testing exp.,1)} | 2 |
| Check mile. | {(Coordinator,1)} | 1 |
| Validate syst. | {(Manager,1),(Software eng.,1)} | 18 |
| Deploy syst. | {(Manager,1)} | 5 |

Table 2: Activity and temporal requirement sets

**Definition 5 (Resource Requirement Set of an Activity).** A resource requirement set of an activity $a \in A$ is $Q_a \subseteq 2^q$. $Q_a$ contains all different resource requirement sets that allow the activity $a$ to be executed.

For instance, given the requirements in Table 2, we now see that *Study requirements* can be executed by one manager *or* by a team of one manager and one coordinator.

On the other hand, data related to the execution of the process (e.g. when and by whom each activity instance is executed) is usually stored in event logs.

**Definition 6 (Event Log).** An event log is a 6-tuple $\mathcal{L} = (E, \varepsilon, \alpha, \varrho, \tau, \mathcal{T})$, where:

- $E = \{e_1, e_2, ..., e_n\}$ is the set of *events*,
- $\varepsilon : E \to \{start, complete\}$ assigns the event type to events,
- $\varrho : E \to R$ assigns the resources to events,
- $\alpha : E \to A$ assigns the activities to events,
- $\tau : E \to \mathbb{Z}_{\geq 0}$ assigns a timestamp to events,
- $\mathcal{T} = \{\sigma_1, \sigma_2, ..., \sigma_n\}$ is the set of *traces* (i.e. process instances), where $\sigma_i \in E^*$ is a trace such that time is non-decreasing (i.e. $1 \leq j < k \leq |\sigma|$: $\tau(\sigma(j)) \leq \tau(\sigma(k))$).

The analysis of the event log can provide valuable information, such as the likelihood of different branches taken at the decision points.

**Definition 7 (Temporal Requirement Function).** The temporal requirement function of an activity requirement set $\delta : (A \times Q_a) \to \mathbb{Z}_{\geq 0}$ returns the duration required for an activity $a \in A$ to be executed with the requirement set $q_a \in Q_a$.

Table 2 indicates that the duration of *Study requirements* is 20 time units (TU) in case it is executed only by one manager, and 15 TU when it is allocated to a team with one manager and one coordinator. With this input data, resources can be allocated.

**Definition 8 (Resource Allocation Problem).** Given a conflict-free Petri net system $PNS$ whose activities are in strict partial order (i.e. the precedence relation between activities is irreflexive, transitive and asymmetrical), an RBAC model $O$, an event log $\mathcal{L}$, one activity requirement set and an upper bound on the process makespan (or process cycle time) $u$, the computation of a feasible allocation of resources is $I \subseteq 2^{(2^R \times A \times U \times U)}$, where each activity $a \in A$ is assigned to several resources that comply with the activity's requirement set $\{r_1, r_2, ..., r_n\} \subseteq 2^R$, a start time $s_a \in U$ and a completion time $c_a \in U$.

Only one resource requirement set per activity needs to be satisfied for the execution of the activity. For the formal representation of the resource allocation problem in business processes, we define two binary variables $o_{ras}$ and $o_{q_a a}$, and a starting time $s \in [0, u]$:

$$o_{q_a a} = \begin{cases} 1, & \text{if the resource requirement set } q_a \in Q_a \text{ is selected for the execution} \\ & \text{of the activity } a, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

$$o_{ras} = \begin{cases} 1, & \text{if the resource } r \text{ is allocated to the activity } a \text{ and } a \text{ starts at time } s, \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

Note that if $o_{q_a a} = 1$, the completion of $a$ occurs at time $s + \delta_{(a,q_a)}$. The objective function and the constraints of the model are as follows. For every $r_i, r_j \in R;\ a_m, a_n \in A;\ q_x \in Q_{a_m};\ q_y \in Q_{a_n};\ s_o, s_p \in U;\ U = \{0, 1, ..., u\};\ u \in \mathbb{Z}_{>0}$ the objective is to

$$\text{minimize } max(\bigcup o_{q_{a_z} a_m} \cdot o_{r_i a_m s_o} \cdot (s_o + \delta_{(a_m,q_z)})) \quad (3)$$

where

$$o_{q_x a_m} \cdot o_{q_y a_m} = 0 \qquad q_x \neq q_y \quad (4)$$

$$\sum o_{r_i a_m s_o} \cdot o_{q_x a_m} = n \cdot o_{q_x a_m} \qquad (a_m, l) \in S_{AL},\ (r_i, l) \in S_{RL},\ (l, n) \in q_x \quad (5)$$

$$o_{r_i a_m s_o} . o_{r_j a_m s_p} = 0 \qquad s_o \neq s_p \quad (6)$$

$$o_{r_i a_m s_o} . (s_o + \delta_{(q_x,a_m)}) \leq o_{r_j a_n s_p} . (s_p) \qquad a_m \xrightarrow{a} a_n,\ o_{q_x a_m} = 1 \quad (7)$$

$$o_{r_i a_m s_o} . o_{r_i a_n s_p} = 0 \qquad a_m || a_n,\ o_{q_x a_m} = 1,\ o_{q_y a_n} = 1,\ [s_o, s_o + \delta_{(q_x,a_m)}] \cap [s_p, s_p + \delta_{(q_y,a_n)}] \neq \emptyset \quad (8)$$

As the set of activities that are not followed by another activity are the last activities to be executed, the objective function (3) minimizes the completion time of the activity that has the greatest value. Constraint (4) ensures there is only one activity requirement set selected for each activity. Constraint (5) indicates that an activity must be allocated as many resources as described in each of its potential activity requirement sets (i.e. roles of required resources and their cardinalities are satisfied). Constraint (6) restricts each activity to having only one start time. Constraint (7) secures that no activity is started until all its predecessors are completed. Constraint (8) enforces that no resource is allocated to any parallel pair of activities that have overlapping execution periods.

A *feasible allocation* occurs when $I$ satisfies the constraints (4-8). By taking into account the minimization objective (3), the time optimal allocation $I_{opt}$ is achieved.

## 4 History-aware Dynamic Process Fragmentation

Our approach is based on two steps. First, we compute the probabilities of taking every single decision involved in the decision points of the process by analyzing previous executions stored in an event log. To do so, we define a decision probability function.

**Definition 9 (Decision Probability Set Function).** The decision probability set function $\Psi_{PN,\mathcal{L}} : (P \times T \times \mathbb{Z}_{>0}) \rightarrow 2^{\mathbb{R}_{[0,1]}}$ builds the set of probabilities of transition $t \in T$ (i.e. decision) being fired at a place $p \in P$ (i.e. decision point) for the $n$th time for each trace $\sigma \in \mathcal{T}$ in the event log $\mathcal{L}$, where $p \in P$, $n \in \mathbb{Z}_{>0}$, $P \in PNS$, $T \in PNS$ and

$\mathcal{T} \in \mathcal{L}$. Let $\varphi : (\mathcal{T} \times T \times \mathbb{Z}_{\geq 1}) \to \mathbb{Z}_{\geq 0}$ be the function that returns the number of occurrences of $t \in T$ in $\sigma \in \mathcal{T}$ at its first $i$ position, where:

$$\varphi(\sigma, t, i) = \sum_{j \in \{1..|\sigma|\}} \begin{cases} 1 & \text{for } \alpha(\sigma(j)) = t,\ j \leq i, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

then the decision probability set is calculated as follows:

$$\Psi_{PN,\mathcal{L}}(p, t, n) = \bigcup_{\sigma \in \mathcal{T}} \frac{\sum_{i \in \{1...|\sigma|\}} \begin{cases} 1 & \text{for } \alpha(\sigma(i)) = t',\ t' \in {}^\bullet p, \\ & \varphi(\sigma, t', i) = n,\ \alpha(\sigma(i+1)) = t, \\ 0 & \text{otherwise.} \end{cases}}{\sum_{i \in \{1...|\sigma|\}} \begin{cases} 1 & \text{for } \alpha(\sigma(i)) = t',\ t' \in {}^\bullet p, \\ & \varphi(\sigma, t', i) = n, \\ 0 & \text{otherwise.} \end{cases}} \tag{10}$$

**Definition 10 (Decision Probability Function).** The decision probability function $\psi_{PN,\mathcal{L}} : (P \times T \times \mathbb{Z}_{>0}) \to \mathbb{R}_{[0,1]}$ calculates the mean probability of a decision $t \in T$ being made at a decision point $p \in P$ for the $n$th time via returning the mean value of elements in the set that the function $\Psi_{PN,\mathcal{L}}$ builds.

$$\psi_{PN,\mathcal{L}}(p, t, n) = \frac{\sum_{x \in \Psi_{PN,\mathcal{L}}(p,t,n)} x}{|\Psi_{PN,\mathcal{L}}(p, t, n)|} \tag{11}$$

For instance, in our running example the option $d_1$*: backend change not required* (i.e. $(p_6, t_1)$) is selected in 20% of the cases and $d_2$*: backend change required* (i.e. $(p_6, t_2)$) in 80% of the cases at their first run. Table 3 shows the probabilities computed from the event log over the Petri net in Fig. 2.

Second and lastly, we select a fragment of the business process for resource allocation. The selection of the fragment depends on a *fragmentation threshold* and the computed decision probabilities. The higher the fragmentation threshold, the smaller the fragment of the process selected for allocation (thus hypothetically the lower the

| Decision | $(p, t)$ | $\psi_{PN,\mathcal{L}}(p, t, 1)$ | $\psi_{PN,\mathcal{L}}(p, t, 2)$ | $\psi_{PN,\mathcal{L}}(p, t, 3)$ |
|---|---|---|---|---|
| $d_1$ | $(p_6, t_1)$ | 0.20 | 0.90 | 0.80 |
| $d_2$ | $(p_6, t_2)$ | 0.80 | 0.10 | 0.20 |
| $d_3$ | $(p_{10}, t_3)$ | 0.55 | 0.40 | 0.70 |
| $d_4$ | $(p_{10}, t_4)$ | 0.45 | 0.60 | 0.30 |
| $d_5$ | $(p_{13}, t_{VS})$ | 0.80 | 0.50 | 0.80 |
| $d_6$ | $(p_{13}, t_5)$ | 0.20 | 0.50 | 0.20 |
| $d_7$ | $(p_{14}, t_{DS})$ | 0.90 | 0.55 | n/a |
| $d_8$ | $(p_{14}, t_6)$ | 0.10 | 0.45 | n/a |

Table 3: Decision probabilities

risk of an unfeasible allocation). A process fragment either starts at the starting place of the process and ends at a decision point, or starts at a decision point and ends at another decision point, or starts at a decision point and ends at the end of the process.

**Definition 11 (Dynamic Process Fragmentation (DPF)).** Given a Petri net system $PNS = (P, T, F, \nu, M_0)$ representing a business process, and a fragmentation threshold $\chi \in \mathbb{R}_{[0,1]}$, we generate a process fragment $\phi = (P', T', F', \Sigma, \lambda')$. The process fragmentation function $\gamma_{PN} : (M \times \chi \times \epsilon_f) \rightarrow 2^{F'}$ generates all the arcs of the process fragment $\phi$ with the help of the decision probability function $\psi_{PN,\mathcal{L}}$ and the state transition function $\Omega_{PN}$. The initial value of the input $\epsilon_f$ is always 1 because the starting arc must be in $\phi$. $\gamma_{PN}$ copies the outgoing arcs at places where the given marking indicates a token while checking for the halting condition, which is bound to decision probability values computed by $\psi_{PN,\mathcal{L}}$, arc probability value $\epsilon_f$, and the fragmentation threshold $\chi$. $\Omega_{PN}$ copies the outgoing arcs at transitions while updating the marking $M$ of the Petri net with respect to a given enabled transition $t \in T$ and returning the updated marking back to $\gamma_{PN}$. The copy function $\kappa : (P \cup T) \rightarrow (P' \cup T')$ copies the nodes of Petri net. The counting function $\theta : (T \times \Sigma) \rightarrow \mathbb{Z}_{\geq 0}$ counts the occurrences of a transition $t \in T$ in a firing sequence $\sigma \in \Sigma$.

$$\gamma_{PN}(M, \chi, \epsilon_f) = \bigcup \begin{cases} (\kappa(p), \kappa(t)) \cup \Omega(t, M, \epsilon_f) & \mu(p) = 1,\ |p^\bullet| = 1,\ t \in p^\bullet, \\ (\kappa(p), \kappa(t)) \cup \Omega(t, M, \epsilon_f \cdot \psi_{PN,\mathcal{L}}(p, t, j)) & \mu(p) = 1,\ |p^\bullet| > 1,\ t \in p^\bullet, \\ & \theta(t, \sigma) = i,\ j = i + 1, \\ & \epsilon_f \cdot \psi_{PN,\mathcal{L}}(p, t, j) \geq \chi, \\ \emptyset & \text{otherwise.} \end{cases} \tag{12}$$

$$\Omega_{PN}(t, M, \epsilon_f) = \begin{cases} \bigcup_{p \in t^\bullet} (\kappa(t), \kappa(p)) \cup \gamma_{PN}(M', \epsilon_f) & \forall p' \in {}^\bullet t, \mu(p') = 1, \\ & M \xrightarrow{t} M', \\ \emptyset & \text{otherwise.} \end{cases} \tag{13}$$

$P'$, $T'$ and $F'$ are defined as follows:

- $F' \subseteq (P' \times T') \cup (T' \times P')$, $\kappa(p) \in P'$, $\kappa(t) \in T'$,
- $P \subseteq P'$ and $T \subseteq T'$,
- $\forall p' \in P'$, $|p'^\bullet| = 1$ and $\phi$ is acyclic.

$\phi$ may contain multiple copies of the nodes (places and transitions) in the original Petri net system $PNS$ when $\chi$ has a value closer to the lower limit 0 and the given $PNS$ contains loops. In such a scenario, the fragmentation may include the same looping decision multiple times. Fig. 3 shows resource allocations with DPF on the software development process (cf. Fig. 2) using the resources described in Tables 1 and 2 at run time with $\chi = 0.65$. After the process fragment $\phi_1$ is generated and the resources are allocated, the process execution starts. In $\phi_1$ the looping decision $d_2$ is taken once and the fragment ends at $p_7$ and $p_{10}$. This indicates that two informed assumptions are made at the decision point $p_6$ (i.e. $\psi_{PN,\mathcal{L}}(p_6, t_2, 1) \cdot \psi_{PN,\mathcal{L}}(p_6, t_1, 2) \geq \chi$). At the first ending place $p_{10}$, neither of the decision probabilities is higher than $\chi$ (i.e. $\psi_{PN,\mathcal{L}}(p_{10}, t_3, 1)$ is

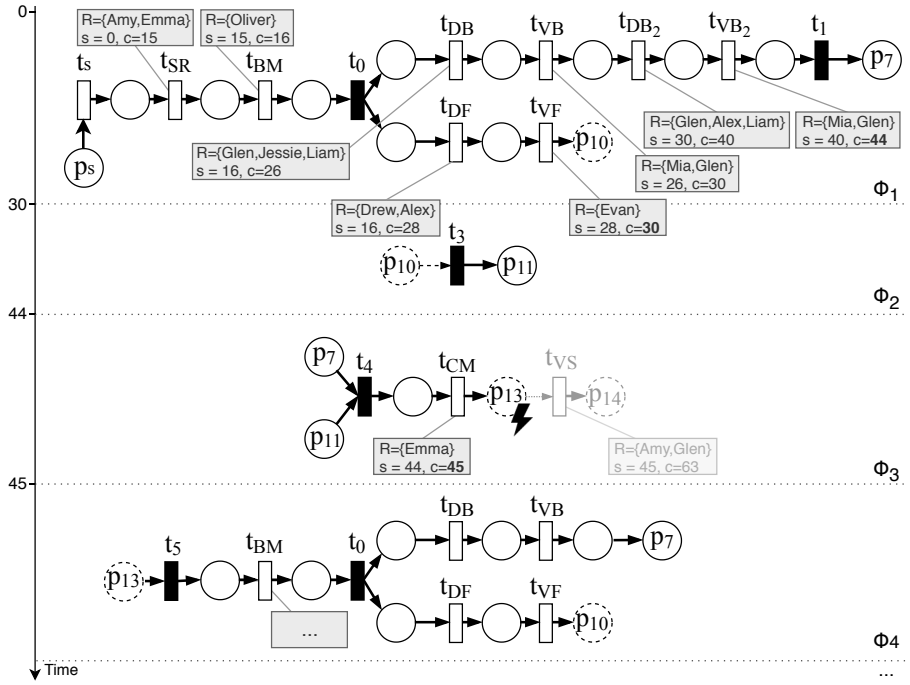



Fig. 3: Process fragments in run-time with a fragmentation threshold of 0.65

0.55 and $\psi_{PN,\mathcal{L}}(p_{10}, t_4, 1)$ is 0.45), hence this branch ends there. The DPF also ends at $p_7$ owing to the fact that the execution of the other parallel branch has not finished and thus $t_4$ cannot be enabled at 30 TU. When *Evan* finishes executing *Verify frontend* at 30 TU, the decision $d_3$ is adopted and the DPF generates the second process fragment $\phi_2$. This fragment ends at $p_{11}$ due to the same reason of $p_7$. At 44 TU, the execution of both parallel branches is finished and the DPF generates the third process fragment $\phi_3$, which ends at $p_{14}$. After the activity *Check Milestone* is executed by *Emma*, $d_6$ is adopted at run time instead of the assumed decision $d_5$ during the DPF. This disruption causes the DPF to generate $\phi_4$ starting from the decision point $p_{13}$ towards the adopted decision $d_6$. This procedure continues until the process execution ends.

## 5 Evaluation

To test the effectiveness of our approach, we have developed a proof-of-concept implementation that lies on the system architecture illustrated in Fig. 4[2]. The components providing input data for our approach are a *Data Repository*, which contains design-time process and organizational data; and *Event logs*, which store traces of past process

[2] With the Fundamental Modeling Concepts (FMC) notation (`www.fmc-modeling.org/`).

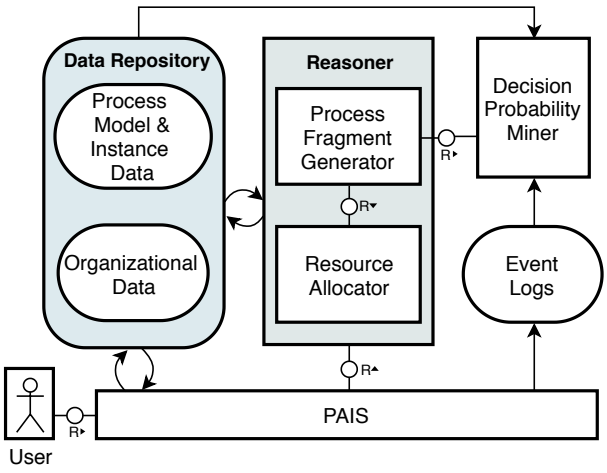



Fig. 4: History-aware Dynamic Process Fragmentation Framework

executions. The other components address the process fragmentation and resource allocation problems and have been implemented as explained next.

*Decision Probability Miner.* This component takes over the implementation of the decision probability function $\psi$ (cf. Equation 11 in Definition 10). It receives a request from the Process Fragment Generator specifying the decision points for which the probability values need to be computed. It takes the process model and the event log as input to derive probability values at these decision points.

*Process Fragment Generator.* This component is the implementation of the $\kappa$, $\Omega$ and $\gamma$ functions described in Definition 11. It receives from the PAIS a request describing the current marking of the Petri net system and the fragmentation threshold value $\chi$, selects an appropriate process fragment $\phi$ and sends a request to the Resource Allocator for performing the allocation of resources on it. Once receiving the response back from the allocator, it returns the optimal allocation $I_{opt}$ back to the PAIS.

*Resource Allocator.* This component performs the resource allocation mechanism as described in Definition 8. It receives a request from the Process Fragment Generator with the process fragment to which the resources are to be allocated. All the other necessary knowledge is retrieved from the Data Repository.

The Resource Allocator and Process Fragment Generator components have been implemented using ASP [3], a declarative (logic-programming-style) paradigm for solving combinatorial search problems. We omit details here due to space limitations but refer to [5] for a brief introduction to ASP and a description of how to encode the resource allocation problem.

As depicted in Fig. 4, both the data sources and the purpose-specific components of the framework interact with a PAIS, which is responsible for process enactment. Note that the framework could be adapted to be used in other environments (e.g. with elastic processes to be executed in the Cloud where cloud resources need to be allocated).

Our method has been evaluated under the hypothesis that the higher the fragmentation threshold value $\chi \in \mathbb{R}_{[0,1]}$ is, the lesser discrepancy occurs between the allocations and run time. To test this hypothesis, we first generate Petri nets with decision points and loops, and label each decision arc $(p, t)$ where $|p^\bullet| > 1$ with random probability values such that the sum of decision probabilities of every $p \in {}^\bullet t$ is 1. Second, we generate the necessary input data for performing the resource allocation task (i.e. Definitions (1-6)) for each Petri net system. Afterwards, we simulate the execution of processes under different fragmentation threshold values and count the number of required reallocations where a low number of reallocation indicates a more robust execution.

We generate the Petri nets using the *Generate block-structured stochastic Petri net* plug-in [15] of the process mining tool ProM [20]. This generator performs a series of random structured insertion operations of new control-flow constructs resulting in a random Petri net given a number of transitions and a degree of parallelism, exclusiveness and cyclicity between the integer values 0 and 100. For our evaluation, we generated 8 Petri nets with 30 activity transitions using this Petri net generator and the properties of the problem instances created are summarized in Table 4. In this table, $id_{PN}$ is the identifier of each different Petri net; $deg_{par}$, $deg_{exc}$ and $deg_{cyc}$ are the input parameters for adjusting the level of parallelism, exclusiveness and cyclicity; and $sym$ is the symbol we use to depict the simulation behavior of Petri nets in Fig. 5. We summarize the properties of problem instances in Table 3 where $id_i$ is the problem instance identifier; $id_{PN}$ is the identifier of the Petri net used (cf. Table 4); and $\chi$ is the fragmentation threshold value used in the simulation environment while dynamically fragmenting the nets and performing the allocation of resources.

| $id_{PN}$ | $deg_{par}$ | $deg_{exc}$ | $deg_{cyc}$ | $sym$ | $id_{PN}$ | $deg_{par}$ | $deg_{exc}$ | $deg_{cyc}$ | $sym$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 45 | 45 | 45 | $\times$ | 5 | 75 | 45 | 45 | $\curlywedge$ |
| 2 | 45 | 75 | 45 | $+$ | 6 | 75 | 75 | 45 | $\prec$ |
| 3 | 45 | 45 | 75 | $\bullet$ | 7 | 75 | 45 | 75 | $\succ$ |
| 4 | 45 | 75 | 75 | $\curlyvee$ | 8 | 75 | 75 | 75 | $\bigstar$ |

Table 4: Properties of Petri net instances

| $id_i$ | $id_{PN}$ | $\chi$ | $id_i$ | $id_{PN}$ | $\chi$ | $id_i$ | $id_{PN}$ | $\chi$ | $id_i$ | $id_{PN}$ | $\chi$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 1 | 0.1 | **11** | 3 | 0.1 | **21** | 5 | 0.1 | **31** | 7 | 0.1 |
| **2** | 1 | 0.25 | **12** | 3 | 0.25 | **22** | 5 | 0.25 | **32** | 7 | 0.25 |
| **3** | 1 | 0.5 | **13** | 3 | 0.5 | **23** | 5 | 0.5 | **33** | 7 | 0.5 |
| **4** | 1 | 0.75 | **14** | 3 | 0.75 | **24** | 5 | 0.75 | **34** | 7 | 0.75 |
| **5** | 1 | 0.9 | **15** | 3 | 0.9 | **25** | 5 | 0.9 | **35** | 7 | 0.9 |
| **6** | 2 | 0.1 | **16** | 4 | 0.1 | **26** | 6 | 0.1 | **36** | 8 | 0.1 |
| **7** | 2 | 0.25 | **17** | 4 | 0.25 | **27** | 6 | 0.25 | **37** | 8 | 0.25 |
| **8** | 2 | 0.5 | **18** | 4 | 0.5 | **28** | 6 | 0.5 | **38** | 8 | 0.5 |
| **9** | 2 | 0.75 | **19** | 4 | 0.75 | **29** | 6 | 0.75 | **39** | 8 | 0.75 |
| **10** | 2 | 0.9 | **20** | 4 | 0.9 | **30** | 6 | 0.9 | **40** | 8 | 0.9 |

Table 5: Properties of problem instances

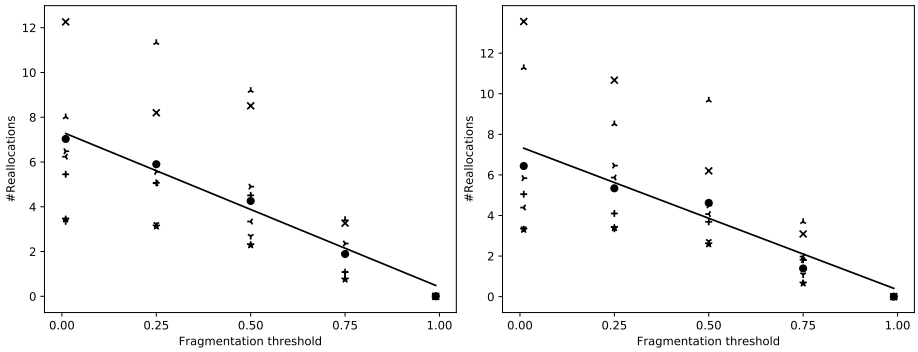



Fig. 5: The fragmentation threshold $\chi$ versus the runtime feasibility of allocations in two different runs

Fig. 5 illustrates two similar results from different simulation runs. The $x$ axis is the fragmentation threshold and the $y$ axis is the number of calls for the resource reallocation mechanism during simulation to be able to repair the existing allocations due to a different decision path is taken at run time than assumed at DPF. The trend lines are almost parallel and they support our hypothesis.

Our DPF implementation has been run on an Ubuntu Linux server (64bit) with a 16 core 2.40 GHz Intel Xeon Processor and 64 GB of RAM. DPF calls consistently took less than a minute and consumed less than 1 GB of memory given Petri nets in Table 4. Our ASP encoding (accompanied with a description) of the Process Fragment Generator, the simulation environment (with visualizations as in Fig. 5) in Python and the problem instances are available at `https://urban.ai.wu.ac.at/~havur/dpf/`.

## 6 Related Work

We next summarize the most representative approaches related to resource allocation in several domains. The evaluation criteria used to compare the approaches include: (i) the risk taken for resource allocation; (ii) whether historical data from event logs is used to more accurately deal with uncertainty; and (iii) whether optimization functions for performance metrics are aimed for or, on the contrary, finding a feasible allocation for the entire process suffices. The results are collected in Table 6, where $N.A.$ indicates that the criterion is non-applicable due to the characteristics of the approach.

| Approach | Risk taken | History-aware? | Optimal solution? |
|---|---|---|---|
| [14] | High | N.A. | No |
| [21, 16] | Low | N.A. | Yes |
| [19, 6] | High | No | No |
| [4, 5] | High | No | Yes |
| DPF (our approach) | Flexible | Yes | Yes |

Table 6: Characteristics of current resource allocation and scheduling methods

Scheduling is concerned with the optimal allocation of scarce resources to tasks over time [10]. The problem has been addressed in several domains, especially in Operations Research, where tasks are typically referred to as "operations" or "steps of a job". In such a domain, surgery scheduling problems, and more specifically the operating room scheduling problem, have been extensively investigated. The most expressive approach in this domain was recently developed by Riise et al. [14]. They propose a model for a "generalised operational surgery scheduling problem" (GOSSP) that can express a wide range of real world surgery scheduling problems as an extension of the multi-mode resource-constrained project scheduling problem with generalized precedence relations (MRCPSP-GPR). Similarly to other approaches in that field, resource allocation is addressed as a two-step problem (assignment and scheduling), and the objective is to minimize the makespan when scheduling several projects (which can contain parallelism but no decision points) at the same time. As an outcome, oftentimes only some projects might be scheduled: those for which all tasks can be allocated. Consequently, feasibility is actually pursued. Besides the surgery scheduling problems, project scheduling in general has been widely investigated [23, 7]. However, due to the inherent differences between projects and business processes (e.g. projects are typically defined to be executed only once and decision points are missing), the problem is approached differently.

In the Business Process Management (BPM) domain, the state-of-the-art on resource allocation does not reach the maturity level of other domains despite the acknowledged importance of the problem [1]. Due to the computational cost that it entails, the existing techniques tend to search either for a feasible solution without applying optimizations [19], or for a local optimal using a greedy approach that might find a feasible but not necessarily a global optimal solution [21]. A framework is proposed in [9] to specify resource requirements where allocation services are independent of respective process models. Otherwise, the existing work has mostly relied on Petri nets. Van der Aalst [21] introduced a timed Petri net based scheduling approach considering activities, resources and temporal constraints. However, modeling this information for multiple process instances leads to very large Petri nets. Rozinat et al. [16] used Coloured Petri nets (CPNs) to overcome the problems encountered in timed Petri nets. In CPNs, classes and guards can be specified to define any kind of constraint. However, the approach is greedy such that resources are allocated to activities as soon as they are available. This may make the allocation problem unsatisfiable or incur in a longer makespan and cost. Several attempts have also been done to implement the problem as a constraint satisfaction problem, considering the business process as a whole for the allocation. For instance, Senkul and Toroslu [19] developed an architecture to specify resource allocation constraints and a Constraint Programming (CP) approach to schedule a workflow according to the constraints defined for the tasks. They aimed at obtaining a feasible rather than an optimal solution, and historical data from previous executions was disregarded. Later on, Heinz and Beck [6] demonstrated that models such as Constraint Integer Programming (CIP) outperform the standard CP formulations. Loops were disregarded in all these approaches. Global optimization is possible at a reasonable computational cost even with complex settings (including loops) using ASP [4, 5]. However, the approaches developed with this formalism had no far con-

sidered historical data in the allocations. The dynamic process fragmentation (DPF) approach presented in this paper bridges the gap currently existing in the BPM domain and advances the state-of-the-art towards more realistic resource allocations.

## 7 Conclusions and Future Work

From this work we conclude that it is possible to allocate resources to process activities taking into account past execution data and the influence of uncertainty in the degree of risk taken when making decisions. Our approach for dynamic process fragmentation allows for a higher flexibility that helps to reduce the reallocation efforts. Furthermore, note that depending on the characteristics of the business process as well as the specific historical values available, the approach can behave as an offline, an online or a *quasi-online* scheduling approach [18]. Future work involves the implementation and testing of the approach in a real setting in connection with a PAIS.

## References


[1] Michael Arias, Eric Rojas, Jorge Munoz-Gama, and Marcos Sepúlveda. A Framework for Recommending Resource Allocation based on Process Mining. In *BPM 2015 Workshops (DeMiMoP)*, pages 458–470, 2015.

[2] Alessandro Colantonio, Roberto Di Pietro, Alberto Ocello, and Nino Vincenzo Verde. A formal framework to elicit roles with business meaning in RBAC systems. In *ACM symposium on Access control models and technologies (SACMAT)*, pages 85–94. ACM, 2009.

[3] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

[4] Giray Havur, Cristina Cabanillas, Jan Mendling, and Axel Polleres. Automated Resource Allocation in Business Processes with Answer Set Programming. In *BPM Workshops (BPI)*, volume 256, pages 191–203. Springer, 2015.

[5] Giray Havur, Cristina Cabanillas, Jan Mendling, and Axel Polleres. Resource Allocation with Dependencies in Business Process Management Systems. In *Int. Conf. on Business Process Management (BPM) - Forum*, volume 260, pages 3–19. Springer, 2016.

[6] Stefan Heinz and Christopher Beck. Solving Resource Allocation/Scheduling Problems with Constraint Integer Programming. In *Constraint Satisfaction Techniques for Planning and Scheduling Problems (COPLAS)*, pages 23–30, 2011.

[7] M.H.A. Hendriks, B. Voeten, and L. Kroep. Human resource allocation in a multi-project R&D environment: Resource capacity allocation and project portfolio planning in practice. *Int. J. of Project Management*, 17(3):181–188, 1999.

[8] Bryan Horling and Victor Lesser. A Survey of Multi-agent Organizational Paradigms. *Knowledge Engineering Review*, 19(4):281–316, 2004.

[9] Sven Ihde, Luise Pufahl, Min-Bin Lin, Asvin Goel, and Mathias Weske. Optimized resource allocations in business process models. In *International Conference on Business Process Management*, pages 55–71. Springer, 2019.

[10] Eugene L. Lawler, Jan Karel Lenstra, Alexander H.G. Rinnooy Kan, and David B. Shmoys. Sequencing and scheduling: Algorithms and complexity. In *Logistics of Production and Inventory*, volume 4 of *Handbooks in Operations Research and Management Science*, pages 445 – 522. Elsevier, 1993.

[11] Niels Lohmann, Eric Verbeek, and Remco Dijkman. Petri Net Transformations for Business Processes - A Survey. *Transactions on Petri Nets and Other Models of Concurrency II*, 2:46–63, 2009.

[12] OMG. BPMN 2.0. Recommendation, OMG, 2011.

[13] Louchka Popova-Zeugmann. Time Petri Nets. In *Time and Petri Nets*, pages 139–140. Springer Berlin Heidelberg, 2013.

[14] Atle Riise, Carlo Mannino, and Edmund K Burke. Modelling and solving generalised operational surgery scheduling problems. *Computers & Operations Research*, 66:1–11, 2016.

[15] Andreas Rogge-Solti. Block-structured stochastic Petri net generator (ProM plugin). `http://www.promtools.org/`, 2014. Accessed: 2019-01-01.

[16] A. Rozinat and R. S. Mans. Mining CPN Models: Discovering Process Models with Data from Event Logs. In *Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN*, pages 57–76, 2006.

[17] Nick Russell, Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and David Edmond. Workflow Resource Patterns: Identification, Representation and Tool Support. In *Conf. on Advanced Inf. Syst. Engineering (CAiSE)*, pages 216–232, 2005.

[18] Ihsan Sabuncuoglu and Selcuk Goren. Hedging production schedules against uncertainty in manufacturing environment with a review of robustness and stability research. *Int. J. of Computer Integrated Manufacturing*, 22(2):138–157, 2009.

[19] Pinar Senkul and Ismail H. Toroslu. An Architecture for Workflow Scheduling Under Resource Allocation Constraints. *Inf. Syst.*, 30(5):399–422, July 2005.

[20] Wil M. P. van der Aalst. *Process Mining - Data Science in Action*. Springer, 2016.

[21] W.M.P. van der Aalst. Petri net based scheduling. *Operations-Research-Spektrum*, 18(4):219–229, 1996.

[22] W.M.P. van der Aalst. *Structural characterizations of sound workflow nets*. Eindhoven University of Technology, Department of Mathematics & Computing Science, 1996.

[23] Jan Weglarz. Project Scheduling with Continuously-Divisible, Doubly Constrained Resources. *Management Science*, 27(9):1040–1053, 1981.

Published in the Proceedings of the 17th Int. Conference on Business Process Management Workshops (BPM 2019 International Workshops), pp. 81-92, Jan 2020, Springer LNBIP vol. 362

# Automated Multi-perspective Process Generation in the Manufacturing Domain*

Giray Havur[1,2], Alois Haselböck[1], and Cristina Cabanillas[2]

[1]Siemens AG Österreich, Corporate Technology, Vienna, Austria
{name.surname}@siemens.com
[2]Vienna University of Economics and Business, Vienna, Austria
{name.surname}@wu.ac.at

**Abstract.** Rapid advances in manufacturing technologies have spurred a tremendous focus on automation and flexibility in smart manufacturing ecosystems. The needs of customers require these ecosystems to be capable of handling product variability in a prompt, reliable and cost-effective way that expose a high degree of flexibility. A critical bottleneck in addressing product variability in a factory is the manual design of manufacturing processes for new products that heavily relies on the domain experts. This is not only a tedious and time-consuming task but also error-prone. Our method supports the domain experts by generating manufacturing processes for the new products by learning the manufacturing knowledge from the existent processes that are designed for similar products to the new products. We have successfully applied our approach in the gas turbine manufacturing domain, which resulted in a significant decrease of time and effort and an increase of quality in the final process design.

**Keywords:** Automated process generation, graph-based process generation, resource assignment, manufacturing process.

## 1 Introduction

With the surging demand for individualized products, the need for flexible production facilities and intelligent manufacturing infrastructures is also increasing. The design of manufacturing processes becomes complex when the products consist of many different parts. The manufacturing industry is still heavily dependent on domain experts to design processes for assembly and production. The search for a feasible process is usually conducted manually by these experts starting with solutions based on analytical calculations and past experiences. The resultant processes are iteratively improved by making changes in the tasks, control-flow and assigned resources. In such a setting, process design becomes a tedious task which is prone to human errors and can lead to sub-optimal processes and a waste of valuable resources. This task also requires a lot of effort when the manufacturing knowledge must be collected from discussions with experts from different disciplines.

* Funded by the Austrian Science Fund (FWF) Elise Richter programme under agreement V 569-N31 (PRAIS).

Many methods, techniques and tools have been developed to support process generation and decision making depending on the structure, availability, and values of product data [1, 2, 3, 4, 5]. However, most of them focus on informational products (where the emphasis is on the collection, processing and aggregation of information), and the automated generation of manufacturing processes for new physical products from the existing processes remains as a challenge.

To assist the domain experts with such a cumbersome task, we have developed a multi-perspective statistical method for generating manufacturing processes from the already existing processes for products which share similar features with the new product. The *functional* perspective is linked with selecting the relevant tasks to the new product, the *behavioral* perspective orders these tasks (i.e. the control flow is implemented), and the *organizational* perspective assigns the necessary resources to the tasks of the generated processes [6]. Therefore, the domain experts are provided with multi-perspective process alternatives that can be reviewed, validated and deployed. We have implemented and tested the approach in a real environment from the gas turbine manufacturing domain. Not only the time and potential cost of the design of the new manufacturing process were saved but also better quality processes happened to be designed following this software-aided approach.

The paper is structured as follows. Section 2 defines concepts needed to understand our approach. Section 3 introduces our process generation method and describes an example scenario that illustrates the method in action. Section 4 briefly describes how the approach has been validated. Section 5 provides a description of the state-of-the-art on related research. Finally, Section 6 outlines the conclusions drawn from the work along with its limitations and potential future steps.

## 2 Background

For the sake of understanding, in the following we briefly define the concepts (stemming from the manufacturing domain) that are involved in our process generation method:

**Definition 1 (Product).** A product $p$ is a finished good that is manufactured. It consists of parts (materials) that can be described by features. The product parts are usually represented as a Bill of Materials (BoM). Such a product could be the result of a manual product design activity.

**Definition 2 (Task).** A task $t$ is a discrete step in the manufacturing process of a product $p$ such as a production operation acting on one or a set of input materials (e.g. drilling a hole into a workpiece or inserting a module into a slot of a base plate).

**Definition 3 (Resource).** A resource $r$ is assigned to the task $t$ for enabling its execution. It can be a production or transportation equipment, like a robot, a computer numerical control (CNC) machine, a 3D printer, or a conveyor belt. Human workers are modeled as resources as well.

**Definition 4 (Manufacturing Process).** A manufacturing process $q$ is a sequence of tasks for manufacturing a product $p$. It is a directed path graph[1] where the vertices

[1] A *path graph* is a graph in which the nodes can be listed in an order $t_1, t_2, \ldots, t_n$ such that the edges are $(t_i, t_{i+1})$ where $i = 1, 2, \ldots, n-1$.

represent the tasks and the edges represent the control flow for manufacturing a product starting from raw or input materials. $q = (T, E)$, where $t_1, t_2, ..., t_n \in T$ are tasks, and each edge $(t_i, t_j) \in E$ represents a direct precedence relation between two tasks, meaning that task $t_j$ starts after task $t_i$ ends. The function $\gamma : (Q \times T) \rightarrow 2^R$ maps each task in a process to a set of possible resources that can execute the task.

**Definition 5 (Feature).** Each product is described by a set of features. Let $F$ be the set of features of the already manufactured products. In the descriptions below, we use the following mappings:

$\phi_t : T \rightarrow 2^F$ maps a task $t$ to the set of features that the task contributes to; $\phi_t(t) \subseteq F$.
$\phi_q : Q \rightarrow 2^F$ maps a process $q = (T, E)$ to the set of features that all of its tasks $T = \{t_1, t_2, \ldots t_n\}$ contribute to; $\phi_q(q) = \bigcup_{i=1..n} \phi_t(t_i)$.

A (product) feature is a property of the final product from the customer's point of view. For instance, a metal workpiece could be *hardened*, *colored*, or *perforated*. We assume that the mapping from tasks to features is labelled manually by domain experts or learned automatically by process mining tools [7] in the manufacturing ecosystem.

The problem space of producing a new product is defined by a set of features, while the manufacturing process with assigned resources represent the solution space. In our context, the relationship between features and tasks is especially important: a task *contributes to* the realization of one or more features. More than one task are typically involved in the realization of a feature.

Therefore, given a new product design *BoM*, a set of features *F*, a set of tasks *T*, and a set of resources *R* extracted from existing processes, the **Process Generation Problem** can be divided into the following sub-problems:

- *Task Selection Problem:* Find a subset $T'$ of the set of tasks T ($T' \subseteq T$) needed for manufacturing the new product design.
- *Process Sequencing Problem:* Find a total ordering relation on $T'$.
- *Resource Assignment Problem:* For each task of $T'$, find a resource that is able to perform this task.

A solution to the process generation problem should fulfill several quality criteria. First of all, a solution must be *technically feasible*, satisfying technical constraints (e.g. the transport of materials from one resource to the resource of the subsequent process must be possible, and a drilling machine for wood can only drill holes into wooden workpieces, not into metals). Second, a solution must be *effective* (i.e. all features of the new product must be covered by the selected tasks). Lastly, a solution must be *efficient*, which means that it should contain no- or only a minimum- number of tasks that do not contribute to the product features. *Technical feasibility* and *effectiveness* of generated processes depend on the quality of existing processes. The *length-optimality* of generated processes is imposed in the proposed method with an optimization function.

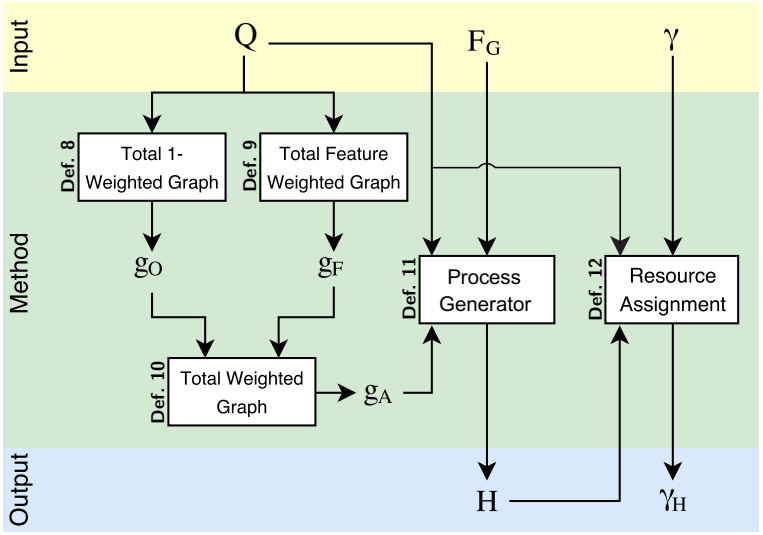



Fig. 1: Technical Features Diagram.

## 3 Automated Multi-perspective Process Generation

We start explaining our work with an overview of the technical features. We refer the reader to Fig. 1 for an overview of input-output relations of the definitions while reading this section. Our major component, the *Process Generator*, takes a set of manufacturing processes $Q = \{q_1, q_2, ..., q_n\}$ of products already manufactured, a set of (goal) features $F_G$ of a new product $p$, and a total weighted graph $g_A$ which contains the task and weights learned from the existing processes and features. A solution to a process generation problem is a set of alternative processes $H = \{h_1, h_2, ..., h_m\}$, each of which is a process candidate for manufacturing the new product $p$ with features $F_G$. On the other hand, the *Resource Assignment* component requires $Q$, $H$, and a function $\gamma$ that maps the tasks in the process set $Q$ to their previously assigned resources. The solution to the resource assignment problem is a function $\gamma_H$, which learns the preferences of resources from $\gamma$ and provides a set of resource-probability pairs for each task in $H$.

We motivate our approach with an example scenario for metalware production derived from the requirements from the gas turbine domain shown in Fig. 2. A metalware factory manufactures three different products: *Steel Pot* (Fig. 2.a), *Gear* (Fig. 2.b), and *File Organizer* (Fig. 2.c). They would like to automatically find out how to produce a *Drainer* (Fig. 2.d) by using the existing manufacturing knowledge. A process details this knowledge into a manufacturing process for each product (e.g. the process for producing *Steel Pot* $q_{sp} = (T_{sp}, E_{sp})$). Note that each task $t$ is remarked by initials of its label (e.g. $t_{ld}$ corresponds to the task ***L**aser **D**rilling*). The task set $T_{sp}$ consists of the tasks $t_s$, $t_{pr}$, $t_{ld}$, $t_w$, and $t_{qc}$. The edge set $E_{sp}$ consists of the edges $(t_s, t_{pr})$, $(t_{pr}, t_{ld})$, $(t_{ld}, t_w)$, and $(t_w, t_{qc})$. The set of resources that can execute each task is shown below the task, e.g. $\gamma(q_{sp}, t_s) = \{r_1, r_2, r_4\}$. We assume that the tasks in the existing processes are

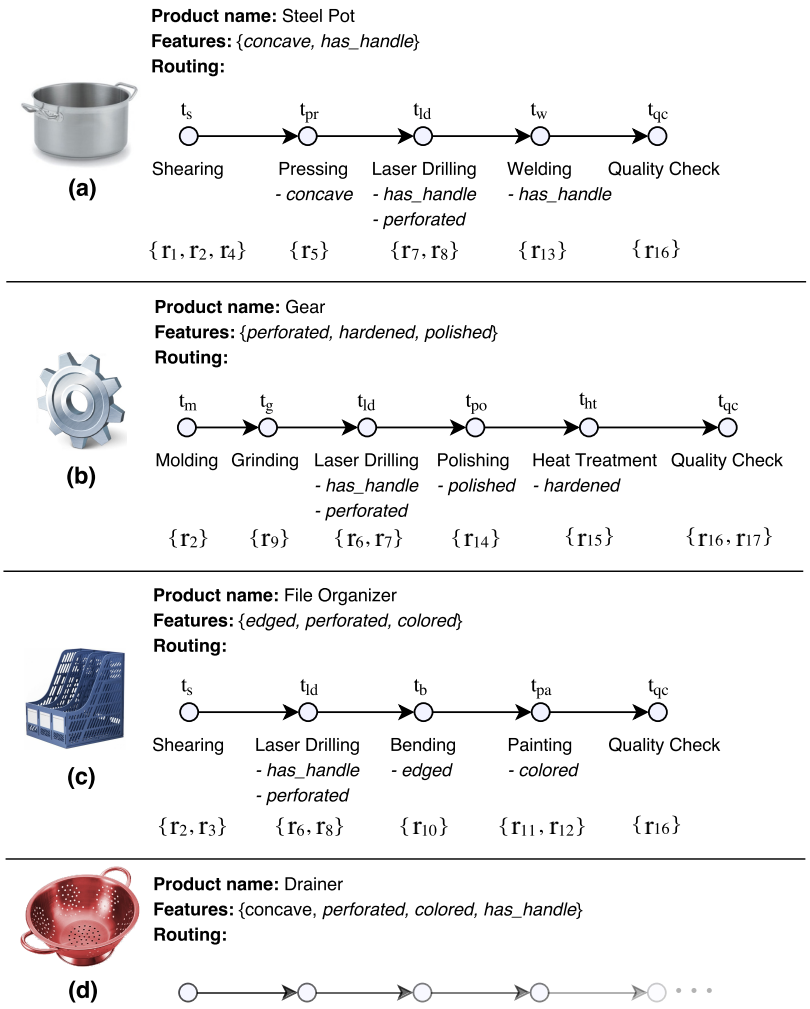



Fig. 2: The products being produced in the metal factory with their corresponding manufacturing processes: (a) Steel Pot with its process $q_{sp}$, (b) Gear with its process $q_g$, (c) File Organizer with its process $q_{fo}$, and the goal product (d) Drainer.

labelled by the features that they contribute to. These features are listed beneath the tasks (e.g. $\phi_t(t_{ld}) = \{has\_handle, perforated\}$). The feature set of the process $q_{sp}$ is $\phi_q(q_{sp}) = \{concave, has\_handle\}$. The feature set of the goal product *Drainer* is $F_G = \{concave, perforated, colored, has\_handle\}$.

**Definition 6 (Task Filtering by Features).** A function $l_t : 2^T \rightarrow 2^T$ selects all tasks of a set of tasks $T$ that contribute to at least one feature of the goal features $F_G$:

$$l_t(T) = \{t \in T \mid \phi_t(t) \cap F_G \neq \emptyset\}$$

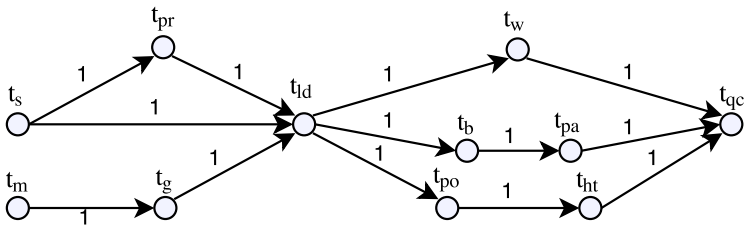



Fig. 3: Total 1-Weighted Graph.

**Definition 7 (Edge Filtering by Features).** A function $l_e : 2^E \rightarrow 2^E$ selects all edges of a process $(T, E)$ whose tasks contribute to at least one feature of the goal features $F_G$:

$$l_e(E) = \{e \in E \mid \phi_t(t_i) \cup \phi_t(t_j) \cap F_G \neq \emptyset \text{ where } e = (t_i, t_j)\}$$

**Definition 8 (Total 1-Weighted Graph).** A total 1-weighted graph $g_O = (T_O, E_O, w_O)$ is derived from a set of processes $Q = \{q_1, q_2, ..., q_n\}$, where $q_i = (T_i, E_i)$, in the following way:

$$T_O = T_1 \cup T_2 \cup ... \cup T_n$$
$$E_O = E_1 \cup E_2 \cup ... \cup E_n$$
$$w_0 : E_O \rightarrow 1 \text{ (all edges have a weight of 1)}$$

Therefore, the total 1-weighted graph is the union of past processes with edge weight of 1. Fig. 3 shows the total 1-weighted graph which is the union of processes in Fig. 2: $q_{sp}$, $q_g$, and $q_{fo}$.

**Definition 9 (Total Feature Weighted Graph).** A total feature weighted graph $g_F = (T_F, E_F, w_F)$ is derived from a set of processes $Q = \{q_1, q_2, ..., q_n\}$, where $q_i = (T_i, E_i)$, in the following way:

$$T_F = \text{tasks of edges } l_e(E_1) \cup l_e(E_2) \cup ... \cup l_e(E_n)$$
$$E_F = l_e(E_1) \cup l_e(E_2) \cup ... \cup l_e(E_n) \cup E_+$$
$$\text{where } E_+ = \left\{(t_i, t_j) \mid t_i, t_j \in l_t(T_F) \wedge \text{ no path}^2 \text{ between } t_i \text{ and } t_j\right\}$$

The weight function $w_F : E \rightarrow \mathbb{Z}_{\geq 0}$ is defined as

$$w_F(e) = \begin{cases} max(1, |(\phi_t(t_1) \cup \phi_t(t_2)) \cap F_G|) \text{ where } e = (t_1, t_2) & \text{if } e \in E_+ \\ |(\phi_t(t_1) \cup \phi_t(t_2)) \cap F_G| \text{ where } e = (t_1, t_2) & \text{otherwise} \end{cases}$$

Following up on our running example, after deriving the total 1-weighted graph we introduce the total feature weighted graph $g_F$ in Fig. 4, which favours the edges with tasks that have overlapping features with $F_G$. Let us consider the edge $(t_{pr}, t_{ld})$.

[2] A *path* in a graph is a sequence of edges which connect a sequence of vertices that are all distinct from one another.

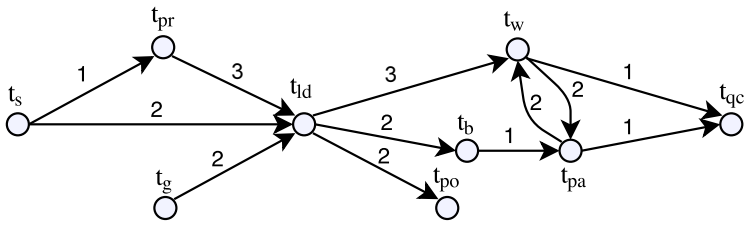



Fig. 4: Total Feature Weighted Graph.

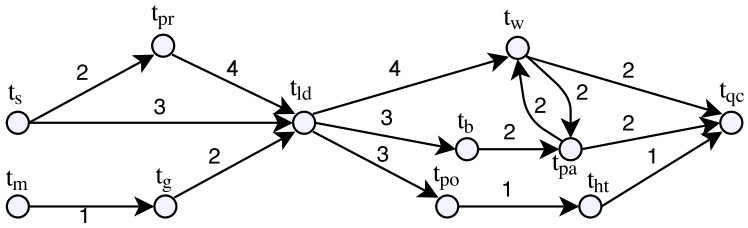



Fig. 5: Total Weighted Graph.

Its weight is $w_F((t_{pr}, t_{ld})) = 3$, because $t_{pr}$ and $t_{ld}$ have 3 features of $F_G$ (*concave*, *perforated*, and *has_handle*). Moreover, the tasks that have overlapping features with $F_G$, but with no existing path in between, are connected bidirectionally. For example, *Welding* $t_w$ and *Painting* $t_{pa}$ have features *has_handle* and *colored* which are in $F_G$, however they are not connected in $g_O$. Therefore, the edges $(t_w, t_{pa})$ and $(t_{pa}, t_w)$ are introduced in $E_+$.

**Definition 10 (Total Weighted Graph).** Let $g_O = (T_O, E_O, w_O)$ and $g_F = (T_F, E_F, w_F)$ be the total 1-weighted graph and the total feature graph corresponding to a set $Q$ of existing processes, respectively. The total weighted graph $g_A = (T_A, E_A, w_A)$ is defined as follows:

$$T_A = T_O$$
$$E_A = E_O \cup E_+$$
$$w_A(e) = \begin{cases} w_O(e) + w_F(e) & \text{if } e \in E_F \\ w_O(e) & \text{if } e \notin E_F \end{cases}$$

The total weighted graph in Fig. 5 is the aggregate graph of $g_O$ and $g_F$ as described above. This graph is the input graph for the process generation.

**Definition 11 (Process Sequencing Problem).**

Let $g_A = (T_A, E_A, w_A)$ be the total weighted graph. Let $T_S \in T_A$ be the set of source tasks, i.e., tasks without incoming edges. Let $T_K \in T_A$ be the set of sink tasks, i.e., tasks without outgoing edges. The goal of process generator is to compute a set of process graphs $H = \{h_1, h_2, ..., h_n\}$ where any $h = (T, E) \in H, T = \{t_i, t_{i+1}, ..., t_j\}$, $E = \{(t_i, t_{i+1}), (t_{i+1}, t_{i+2}), ..., (t_{j-1}, t_j)\}$ has the following properties:

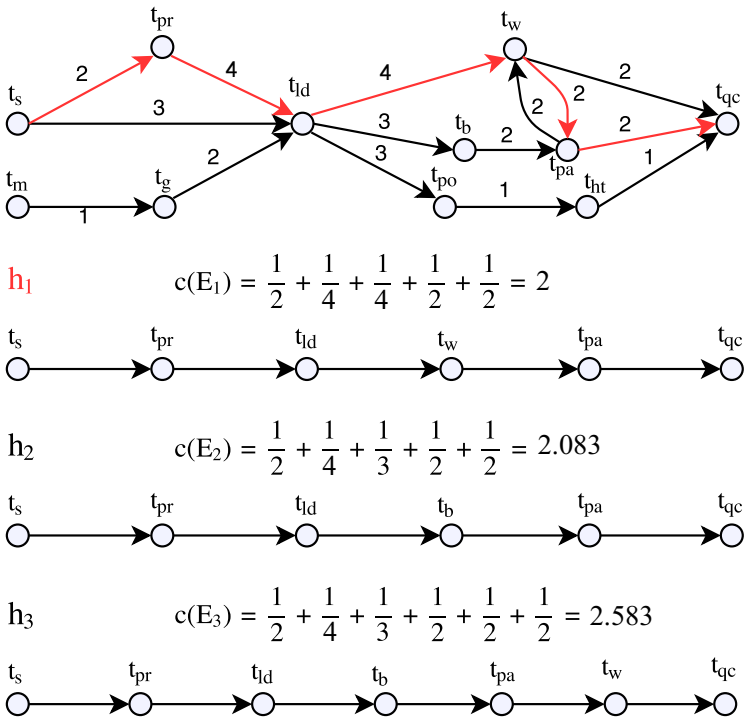



Fig. 6: Generated process alternatives $H = \{h_1, h_2, h_3\}$ for Drainer.

- $h$ is acyclic,
- $h$ starts from a source task ($t_i \in T_S$) and ends at a sink task ($t_j \in T_K$), and
- $\phi_q(h) \supseteq F_G$.

The minimization of an objective function $c(E)$ is desirable for ordering the generated process graphs:

$$c(E) = \sum_{e \in E} \frac{1}{w_A(e)}$$

Fig. 6 shows three alternative generated processes $h_1$, $h_2$, and $h_3$ for manufacturing *Drainer*. The process with the minimum optimization function value $h_1$ is highlighted in the total weighted graph $g_A$. All the generated processes satisfies the properties in Def. 11, namely

- $h_1$, $h_2$, and $h_3$ are path graphs,
- $t_s$ is a source task, and $t_{qc}$ is a sink task, and
- $\phi_q(h_1) \supseteq F_G$, $\phi_q(h_2) \supseteq F_G$, and $\phi_q(h_3) \supseteq F_G$, i.e. the union set of features of each generated process is a superset or equal to $F_G$.

The processes are ranked with respect to the value of objective function, i.e. $c(E_1) < c(E_2) < c(E_3)$.

**Definition 12 (Resource Assignment Problem).** The Resource Assignment Problem is defined by:

- a set of processes $Q = \{q_1, q_2, ..., q_n\}$,
- a function $\gamma : (Q \times T) \rightarrow 2^R$ that maps a task $t$ in a process $q$ to its set of resources $R_{(q,t)}$,
- a set of generated processes $H = \{h_1, h_2, ..., h_m\}$.

A solution to a Resource Assignment Problem $(Q, \gamma, H)$ consists of a function $\gamma_H : T \rightarrow 2^{(R \times [0..1])}$ that maps a task $t \in T$ to its set of resource-probability pair set $\{(r_i, p(r_i, t)), ..., (r_j, p(r_j, t))\}$, e.g. $p(r_i, t)$ reflects how likely the resource $r_i$ is used for executing the task $t$.

In order to obtain $\gamma_H$, we need two auxiliary definitions. First one is the set of all possible resources that can execute the task $t$:

$$R_t^* = \bigcup_{q=(T,E) \in Q} \bigcup_{t \in T} \gamma(q, t)$$

Then, we define the function $p(r, t)$ for returning the probability of resource $r$'s executing task $t$:

$$p(r,t) = \frac{\sum_{q \in Q} \begin{cases} 1/|\gamma(q,t)| & \text{if } q \in \gamma(q,t) \\ 0 & \text{otherwise} \end{cases}}{\sum_{q \in Q} \begin{cases} 1 & \text{if } t \in T \text{ where } q = (T,E) \\ 0 & \text{otherwise} \end{cases}}$$

Finally, $\gamma_H(t)$ is defined as:

$$\gamma_H(t) = \bigcup_{r \in R_t^*} (r, p(r, t))$$

In our running example, after generating three alternative processes for the goal product *drainer*, we need to assign resources to tasks in the candidate processes so that they can become executable. For this purpose, we use the function $\gamma_H(t)$ which computes the probability of execution for each resource that has already executed the task $t$ in the past processes. For example, the task $t_s$ has been executed by the resources $\{r_1, r_2, r_4\}$ in $q_{sp}$, and by $\{r_2, r_3\}$ in $q_{fo}$. The resources $\{r_1, r_2, r_3, r_4\}$ are therefore feasible resources for executing $t_s$. However, we also know that $r_2$ has been assigned for executing $t_s$ in both past processes, which should make $r_2$ a better option than others. We incorporate this knowledge by deriving probabilities from the assignment frequencies of these resources to tasks. For example, the resource assignment for $t_s$ is

$\gamma_H(t_s) = \{(r_1, p(r_1, t_s)), (r_2, p(r_2, t_s)), (r_3, p(r_3, t_s)), (r_4, p(r_4, t_s))\}$ where

$$p(r_1, t_s) = \frac{\frac{1}{3} + 0 + 0}{1 + 0 + 1} = 0.166 \quad p(r_2, t_s) = \frac{\frac{1}{3} + 0 + \frac{1}{2}}{1 + 0 + 1} = 0.416$$
$$p(r_3, t_s) = \frac{0 + 0 + \frac{1}{2}}{1 + 0 + 1} = 0.25 \quad p(r_4, t_s) = \frac{\frac{1}{3} + 0 + 0}{1 + 0 + 1} = 0.166$$

Hence, $\gamma_H(t_s) = \{(r_1, 0.166), (r_2, 0.416), (r_3, 0.25), (r_4, 0.166)\}$. This means $r_2$ is a good candidate for executing $t_s$, which is followed by $r_3$, and by equally $r_1$ and $r_4$. For assigning resources to the generated processes in Fig. 6, we need to compute $\gamma_H(t_s)$, $\gamma_H(t_{pr})$, $\gamma_H(t_{ld})$, $\gamma_H(t_w)$, $\gamma_H(t_{pa})$, $\gamma_H(t_{qc})$, and $\gamma_H(t_b)$.

## 4 Validation

We have successfully applied our approach in the turbine manufacturing domain to automatically generate processes for new gas turbine types that have specific modifications upon the existing gas turbine production processes. These modifications in product specifications (e.g. blades and vanes of different sizes) require different sequences of tasks (i.e. assembly and logistics) that are executed by specific resources. In this application, the process base (i.e. existing set of processes) has 173 existing processes, and a typical process has more than 30 tasks. We have generated 10 processes for different turbine configurations which have been reviewed by a domain expert. Afterwards, these generated processes have been used as input in the manufacturing simulation and validation tool Tecnomatix[3]. This simulation tool has also validated the feasibility of the generated processes from the functional, behavioral and organizational perspectives. The domain expert has experienced a considerable time saving in by employing our method in final process design task. Due to company policy, further comments in this section are omitted. Please contact the authors in case of any requests for instance details of the validation.

## 5 Related Work

Several methods have been proposed and developed to automatically explore and evaluate a range of different product designs, their options and parameters [8, 9]. These methods ensure feasibility of the resulting products (i.e. their configuration must be consistent and complete regarding the constraints and requirements imposed in the product specifications).

Efforts have been made in modeling decision structures and dependencies to steer the execution of processes [4]. Such efforts are extended towards including product data [3] and for configuring processes [10] to offer flexibility in run-time process execution.

There are general process generation approaches which are taking into account data-flow elements of a process and the part-of relationships of products (i.e. BoMs) by

[3] https://www.plm.automation.siemens.com/global/de/products/tecnomatix/logistics-material-flow-simulation.html

ensuring an executable ordering of tasks that describe how to produce a goal product [1, 2].

An alternative approach in automated process design stems from the research in Product Lifecycle Management (PLM). PLM tools support the modeling and representation of product portfolios. A product portfolio (or *product line*) stands for a whole set of product instances. A product instance (i.e. an individualized product) is usually derived from the product portfolio by a *product configuration* step with the help of a configuration tool. Recent research extends this product configuration by encompassing also the needed production processes for manufacturing the individualized product [11, 12, 13, 14]. The main differences to the approach proposed in this work are: (i) Production processes are created only for instances of the product portfolio but not for completely new product designs, represented by a new vector of features; and (ii) Production operations for the different product parts are manually modeled and represented in the PLM tool or an extra knowledge base is used in the production workflow generation tool, whereas in our case such knowledge is derived from existing production processes of products.

## 6 Conclusions

We have introduced a novel method which derives a statistical model from the existing production processes to generate new processes with resource assignments for new products. Our method can be employed as a building block in a smart production ecosystem aiming at flexibility and automation. It has been validated in an industrial gas turbine production setting and proven to facilitate the process design efforts for new gas turbines.

As future work we plan to apply our method in other manufacturing settings (e.g. mobility and medical hardware) and also generalize the method towards the service operations to explore the capabilities of our method in a new domain.

## References


[1] Wil MP van der Aalst. On the automatic generation of workflow processes based on product structures. *Computers in Industry*, 39(2):97–111, 1999.

[2] Han van der Aa, Hajo A Reijers, and Irene Vanderfeesten. Composing workflow activities on the basis of data-flow structures. In *Business Process Management*, pages 275–282. Springer, 2013.

[3] Irene Vanderfeesten, Hajo A Reijers, and Wil MP Van der Aalst. Product-based workflow support. *Information Systems*, 36(2):517–535, 2011.

[4] Feng Wu, Laura Priscilla, Mingji Gao, Filip Caron, Willem De Roover, and Jan Vanthienen. Modeling decision structures and dependencies. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 525–533. Springer, 2012.

[5] Krzysztof Kluza and Grzegorz J Nalepa. Automatic generation of business process models based on attribute relationship diagrams. In *International Conference on Business Process Management*, pages 185–197. Springer, 2013.

[6] Cristina Cabanillas. Process- and Resource-Aware Information Systems. In *Int. Conf. on Enterprise Distributed Object Computing (EDOC)*, pages 1–10, 2016.

[7] Wil Van Der Aalst. *Process mining: discovery, conformance and enhancement of business processes*, volume 2. Springer, 2011.

[8] Luisa Paraguai, Heloisa Candello, and Paulo Costa. Collaborative system for generative design: Manipulating parameters, generating alternatives. In *Design, User Experience, and Usability: Theory, Methodology, and Management - 6th International Conference, DUXU 2017, Held as Part of HCI International 2017, Vancouver, BC, Canada, July 9-14, 2017, Proceedings, Part I*, pages 727–739, 2017.

[9] Deepak Dhungana, Andreas Falkner, Alois Haselböck, and Richard Taupe. Enabling integrated product and factory configuration in smart production ecosystems. In *Proceedings of the 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Vienna (Austria), 2017.

[10] Luisa Parody, María Gómez-López, Angel Varela-Vaca, and Rafael Gasca. Business process configuration according to data dependency specification. *Applied Sciences*, 8(10):2008, 2018.

[11] Lavindra de Silva, Paolo Felli, Jack C Chaplin, Brian Logan, David Sanderson, and Svetan Ratchev. Realisability of production recipes. In *European Conference on Artificial Intelligence*, 2016.

[12] Deepak Dhungana, Alois Haselböck, and Richard Taupe. A marketplace for smart production ecosystems. In *Mass Customization & Personalization Conference*, 2017.

[13] Deepak Dhungana, Andreas Falkner, Alois Haselböck, and Herwig Schreiner. Smart factory product lines: a configuration perspective on smart production ecosystems. In *Proceedings of the 19th International Conference on Software Product Line*, pages 201–210. ACM, 2015.

[14] Dario Campagna and Andrea Formisano. Product and production process modeling and configuration. *Fundam. Inform.*, 124(4):403–425, 2013.

# Conclusion and the Way Ahead

In this cumulative thesis, we presented the following contributions for the research questions (1-3) which have been published in [1–9]. The first conceptualization and implementation of RABP was given in [1]. Further extensions to RABP towards advanced resource and time management was described in [2]. Our evaluations show that RABP is challenging from a computational perspective, and finding a feasible solution to RABP with makespan minimization was classified as NP-Hard (See Appendix B). For encoding RABP and solving instances of real-life sizes, KRR formalisms and their solvers were tested against each other to provide insights into selecting the most suitable formalism for RABP. ASP and CP solutions to RABP were compared in [3], and a configurable ASP systems benchmark environment that generates RABP instances and tests the performance of ASP systems was devised and implemented in [4, 10]. Our efforts to encode RABP in different formalisms proved the ASP encoding to be compact, human-readable, and easy to extend. In contrast, the CP implementation has better computational performance (i.e., time and memory usage) in solving RABP instances.

We studied several methods for RABP support in practice. Performing RABP in engineering domains requires following the norms and criteria to reduce the incidence of failures in both design-time (i.e., planning stage of projects) and run-time (i.e., execution stage) of BPs. For example, the RAMS (Reliability[1], Availability[2], Maintainability[3], and Safety[4]) disciplines [11] ensures that a product, process, or system will carry out its intended function.

Our first demo addressing RABP for safety-critical engineering projects was implemented as a Camunda BPMS module [6]. An extended version of this work further enhanced *safety* aspects in BP executions (i.e., monitoring of the executions, adaptation to resource- and process-related changes, and their documentation) [7]. Moreover, we devised a history-aware process fragmentation method for increasing the *reliability* of RABP in [9] and detailed a stochastic prediction method [8] to forecast the *availability* of resources from the BP control flows and the organizational structures. Lastly, we explored the manufacturing domain: a multi-perspective (functional, behavioral, and organizational) process generation method [12] was developed to support the production of new products by reducing the manual work spent on the design of production processes, hence improving the *maintainability* of production processes.

Our contributions suggest several research directions that can be investigated in the future:

[1] Reliability is the ability to perform a particular function.

[2] Availability is the ability to keep a functioning state.

[3] Maintainability is the ability to be timely and easily maintained (including servicing, inspection and check, repair and/or modification).

[4] Safety is the ability not to harm people, the environment, or any assets.

**An RABP description language:** A more standardized language for generally describing RABP could be developed to represent RABP instances (BPs, organizational models, temporal models, constraints, and objectives). Well-defined translations would allow RABP problem instances to be seamlessly solved by multiple formalisms and their solvers. Among the related work, a road map for semantic BPM was laid in [13]. In this study, the representational requirements of BPM are investigated, and a set of ontologies for representing BPs and organizational models are proposed. A current key limitation for a generalized RABP language is the lack of alignment in the developed ontologies/standards: several process-related ontologies (e.g., BPMN and Petri nets) were developed to represent activities and control flows [14–16]; various resource descriptions were developed to cover mainly human resources [17–21]. In [7], we proposed a resource ontology including human and non-human resources (e.g. tools and infrastructure), resource-related compliance constraints, and the underpinning properties that connect resources with a Petri net ontology as a initial effort for a generalized RABP description.

**Hierarchical RABP:** Specific BPM use cases might require further extensions in the types of constraints, objectives, and execution modes. For example, in a BPM setting where a multi-level organization is defined as a combination of hierarchical sub-organizations (e.g., hierarchy models in ISA-95 [22]), RABP must deal with the multi-level priorities and objectives to find an optimal schedule with the allocation of resources. The stipulated objectives and constraints propagated from upper hierarchy BPs (e.g., management processes) must be mediated through (semi-)automated negotiations towards lower-level BPs (e.g., logistics processes) [23]. Furthermore, BPs at different levels may compete or cooperate for resources by imposing their objectives in such a scenario, where multi-objective optimizations on different levels of the process hierarchy would need to be investigated, in order to align (possibly competing) goals on a strategic level.

**RABP for robust BP executions:** BP execution failures can occur due to unexpected changes in resource availability and on-the-fly modifications in BPs. Approaches to address robustness can be put under three main categories: avoiding future failures via prospective methods (e.g., forecasting utilization of resources [8]), improving resistance to failures via variability/flexibility by design (e.g., adaptive resource allocation [24] and adaptive business processes [25]), and recovering from failures via retrospective methods (e.g., resource re-allocation [7]). Futher refining and aligning such RABP methods that are resilient to unforeseen changes shall enhance the overall robustness of BP executions.

**RABP dashboard for BPMS:** Integrating interactive RABP dashboards in BPMSs would simplify setting up and running RABP instances [20] (e.g., listing resource preferences, modifying BP-dependent objectives, and con-

figuring automated solvers). Moreover, resource-related data visualizations could help users monitor the resources (e.g., historical, current, and future availability of resources, cost/progress trends, and deadline risk estimations) and take informed action regarding BP executions when necessary. Besides these functionalities, a dashboard could also facilitate comparing and contrasting alternative RABP solutions bound to different objectives (e.g., with minimum duration vs. minimum cost).

**Large-scale RABP solving methods and their analysis:** In this thesis, we have investigated exact RABP methods for computing global-optimal resource allocations. Such methods guarantee the best solution; however, as we have seen, finding the optimal solution to RABP is computationally expensive and therefore, could be prohibitively time consuming. It is important to balance the required time to generate a feasible RABP solution with the resulting quality of the allocation (i.e., closeness to a global-optimal solution). To address large-scale domain-specific resource allocation needs, approximate RABP implementations can use heuristics (e.g., [26]), metaheuristics (e.g., [27]) and/or machine learning methods (e.g., [28]) to deliver solutions in less time than exact methods. Continued research on such (meta-)heuristics and machine learning models appears fully justified due to the wide range of RABP requirements. Moreover, to our best knowledge, no publication experimentally measures the applicability and performance of these methods in a selected subset of BPM use cases.

**Bridging the gap between BPM and scheduling:** Most research in the scheduling domain has been driven by project management [29] and manufacturing systems [30]. For half a century, great effort has been devoted to identifying and studying key problems in scheduling: among others, resource types (e.g., renewable resources, non-renewable resources, etc.), running schemes (e.g., calling the scheduling method once at the beginning of the operation/project vs. multiple-times at fixed time intervals vs. event-driven), scheduling environments (e.g., fully specified vs. partially specified), processing of the input data (e.g., offline vs. online), adaptation methods (e.g., repair planning), and implementation techniques (e.g., rule-based, machine learning, heuristic, and linear-programming). Moreover, several categorization schemes [31–33] have been developed to describe a great variety of these problems. The alignment of concepts and methods between the BPM and scheduling research is invaluable to moving the RABP-related knowledge forward, and may cross-fertilize both fields, BPM and scheduling alike, where we hope that the works presented in this thesis have provided some starting points.

This thesis contains three appendices: in Appendix A, we demonstrate refined experiments where we show the convergence to optimal solutions in five selected RABP instances from [4]; in Appendix B, we provide the proof of the conjecture on the computational complexity of RABP (i.e., RABP with makespan optimization

is NP-Hard) in [4]; and in Appendix C, we compare two different implementations of RABP in ASP and CP.

## References


[1] Giray Havur, Cristina Cabanillas, Jan Mendling, and Axel Polleres. Automated resource allocation in business processes with answer set programming. In *Business Process Management Workshops: BPM 2015, 13th International Workshops, Innsbruck, Austria, August 31 – September 3, 2015, Revised Papers*, pages 191–203, Cham, 2016. Springer International Publishing.

[2] Giray Havur, Cristina Cabanillas, Jan Mendling, and Axel Polleres. Resource allocation with dependencies in business process management systems. In Marcello La Rosa, Peter Loos, and Oscar Pastor, editors, *Business Process Management Forum*, pages 3–19, Cham, 2016. Springer International Publishing.

[3] Giray Havur. A comparison of ASP and CP solutions for resource allocation in business processes. Technical report, Working Papers on Information Systems, 2022.

[4] Giray Havur, Cristina Cabanillas, and Axel Polleres. Benchmarking answer set programming systems for resource allocation in business processes. *Expert Systems with Applications*, 205:117599, 2022.

[5] Saimir Bala, Giray Havur, Simon Sperl, Simon Steyskal, Alois Haselböck, Jan Mendling, and Axel Polleres. SHAPEworks: A BPMS Extension for Complex Process Management. In *BPM Demos*, 2016.

[6] Saimir Bala, Giray Havur, Simon Sperl, Simon Steyskal, Alois Haselböck, Jan Mendling, and Axel Polleres. SHAPEworks: A BPMS extension for complex process management. In *Proceedings of the BPM Demo Track 2016 Co-located with the 14th International Conference on Business Process Management (BPM 2016), Rio de Janeiro, Brazil, September 21, 2016*, volume 1789 of *CEUR Workshop Proceedings*, pages 50–55. CEUR-WS.org, 2016.

[7] Saimir Bala, Cristina Cabanillas, Alois Haselböck, Giray Havur, Jan Mendling, Axel Polleres, Simon Sperl, and Simon Steyskal. A framework for safety-critical process management in engineering projects. In *Data-Driven Process Discovery and Analysis - 5th IFIP WG 2.6 International Symposium, SIMPDA 2015, Vienna, Austria, December 9-11, 2015, Revised Selected Papers*, volume 244 of *Lecture Notes in Business Information Processing*, pages 1–27. Springer, 2015.

[8] Simon Sperl, Giray Havur, Simon Steyskal, Cristina Cabanillas, Axel Polleres, and Alois Haselböck. Resource utilization prediction in decision-intensive business processes. In *Proceedings of the 7th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2017), Neuchâtel, Switzerland, December 6-8, 2017*, volume 2016 of *CEUR Workshop Proceedings*, pages 128–141. CEUR-WS.org, 2017.

[9] Giray Havur and Cristina Cabanillas. History-aware dynamic process fragmentation for risk-aware resource allocation. In *On the Move to Meaningful Internet Systems: OTM 2019 Conferences - Confederated International Conferences: CoopIS, ODBASE, C&TC 2019, Rhodes, Greece, October 21-25, 2019, Proceedings*, volume 11877 of *Lecture Notes in Computer Science*, pages 533–551. Springer, 2019.

[10] Giray Havur, Cristina Cabanillas, and Axel Polleres. BRANCH: An ASP systems benchmark for resource allocation in business processes. In *Proceedings of the Best Dissertation Award, Doctoral Consortium, and Demonstration & Resources Track at BPM 2021*, volume 2973 of *CEUR Workshop Proceedings*, pages 176–180. CEUR-WS.org, 2021.

[11] Alain Villemeur. *Reliability, Availability, Maintainability and Safety Assessment: Method and Techniques*. John Wiley and Sons, 1991.

[12] Giray Havur, Alois Haselböck, and Cristina Cabanillas. Automated multi-perspective process generation in the manufacturing domain. In *Business Process Management Workshops - BPM 2019 International Workshops, Vienna, Austria, September 1-6, 2019, Revised Selected Papers*, volume 362 of *Lecture Notes in Business Information Processing*, pages 81–92. Springer, 2019.

[13] Martin Hepp and Dumitru Roman. An ontology framework for semantic business process management. In *eOrganisation: Service-, Prozess-, Market-Engineering: 8. Internationale Tagung Wirtschaftsinformatik - Band 1, WI 2007, Karlsruhe, Germany, February 28 - March 2, 2007*, pages 423–440. Universitaetsverlag Karlsruhe, 2007.

[14] Christine Natschläger. Towards a BPMN 2.0 ontology. In *Business Process Model and Notation - Third International Workshop, BPMN 2011, Lucerne, Switzerland, November 21-22, 2011. Proceedings*, volume 95 of *Lecture Notes in Business Information Processing*, pages 1–15. Springer, 2011.

[15] Marco Rospocher, Chiara Ghidini, and Luciano Serafini. An ontology for the business process modelling notation. In *Formal Ontology in Information Systems - Proceedings of the Eighth International Conference, FOIS 2014, September, 22-25, 2014, Rio de Janeiro, Brazil*, volume 267 of *Frontiers in Artificial Intelligence and Applications*, pages 133–146. IOS Press, 2014.

[16] Juan C Vidal, Manuel Lama, and A Bugarín. A high-level petri net ontology compatible with pnml. *Petri Net Newsletter*, 71:11–23, 2006.

[17] Cristina Cabanillas, Manuel Resinas, and Antonio Ruiz-Cortés. RAL: A High-Level User-Oriented Resource Assignment Language for Business Processes. In *Business Process Management Workshops (BPD'11)*, pages 50–61, 2011.

[18] Cristina Cabanillas, Manuel Resinas, Jan Mendling, and Antonio Ruiz Cortés. Automated team selection and compliance checking in business processes. In *Proceedings of the 2015 International Conference on Software and System Process, ICSSP 2015, Tallinn, Estonia, August 24 - 26, 2015*, pages 42–51, 2015.

[19] Agata Filipowska, Martin Hepp, Monika Kaczmarek, and Ivan Markovic. Organisational ontology framework for semantic business process management. In *Business Information Systems, 12th International Conference, BIS 2009, Poznan, Poland, April 27-29, 2009. Proceedings*, volume 21 of *Lecture Notes in Business Information Processing*, pages 1–12. Springer, 2009.

[20] Isabelle Linden. Proposals for the integration of interactive dashboards in business process monitoring to support resources allocation decisions. *J. Decis. Syst.*, 23(3):318–332, 2014.

[21] Emna Hachicha and Walid Gaaloul. Towards resource-aware business process development in the cloud. In *29th IEEE International Conference on Advanced Information Networking and Applications, AINA 2015, Gwangju, South Korea, March 24-27, 2015*, pages 761–768. IEEE Computer Society, 2015.

[22] Charlotta Johnsson. Isa 95-how and where can it be applied. *ISA Expo*, pages 1–10, 2004.

[23] Karoline Feigl. Propagation and mediation of resource allocation constraints in multi-level organizations. Department of Information Systems and Operations, Vienna University of Economics and Business, 2020. (Bachelor's Thesis).

[24] Philipp Hoenisch, Stefan Schulte, Schahram Dustdar, and Srikumar Venugopal. Self-adaptive resource allocation for elastic process execution. In *2013 IEEE Sixth International Conference on Cloud Computing, Santa Clara, CA, USA, June 28 - July 3, 2013*, pages 220–227. IEEE Computer Society, 2013.

[25] Riccardo Cognini, Flavio Corradini, Stefania Gnesi, Andrea Polini, and Barbara Re. Research challenges in business process adaptability. In *Symposium on Applied Computing, SAC 2014, Gyeongju, Republic of Korea - March 24 - 28, 2014*, pages 1049–1054. ACM, 2014.

[26] Karl F. Doerner, Walter J. Gutjahr, Gabriele Kotsis, Martin Polaschek, and Christine Strauss. Enriched workflow modelling and stochastic branch-and-bound. *Eur. J. Oper. Res.*, 175(3):1798–1817, 2006.

[27] Felicitas Fabricius, Marco De Bortoli, Maximilian Selmair, Michael Reip, Gerald Steinbauer, and Martin Gebser. Towards asp-based scheduling for industrial transport vehicles. In *Joint Austrian Computer Vision and Robotics Workshop*, 2020.

[28] Kamil Zbikowski, Michal Ostapowicz, and Piotr Gawrysiak. Deep reinforcement learning for resource allocation in business processes. *CoRR*, abs/2104.00541, 2021.

[29] Christoph Schwindt, Jürgen Zimmermann, et al. *Handbook on project management and scheduling vol. 1*. Springer, 2015.

[30] Michael Pinedo. *Scheduling*. Springer, 2015.

[31] Willy Herroelen, Erik Demeulemeester, and Bert De Reyck. *A Classification Scheme for Project Scheduling*, pages 1–26. Springer US, Boston, MA, 1999.

[32] Stijn Van de Vonder, Erik Demeulemeester, and Willy Herroelen. A classification of predictive-reactive project scheduling procedures. *J. Sched.*, 10(3):195–207, 2007.

[33] Peter Brucker, Andreas Drexl, Rolf H. Möhring, Klaus Neumann, and Erwin Pesch. Resource-constrained project scheduling: Notation, classification, models, and methods. *Eur. J. Oper. Res.*, 112(1):3–41, 1999.

# Appendices

# Appendix A

# Makespan Optimization Performance of Five Challenging RABP Instances Solved by ASP Systems (GRINGO+CLASP), (IDLV+CLASP), (GRINGO+WASP), and (IDLV+WASP)

**Author:** Giray Havur

Supplementary material to the journal publication *Benchmarking Answer Set Programming Systems for Resource Allocation in Business Processes* published in the International Journal on Expert Systems with Applications(ESWA), Volume 205, Number 117599, May 2022, Elsevier.

In [1], we define a formalization of the RABP problem, provide a baseline problem encoding in ASP, develop a ready-to-use, configurable benchmark named BRANCH, and present a detailed evaluation that compares four ASP systems comprising combinations of state-of-the-art ASP grounders GRINGO [2] and I-DLV [3], and solvers CLASP [4] and WASP [5].

We rerun the RABP instances 16, 46, 55, 59, and 62 in [1, Section 5] without time and memory limitations per instance. Table 1 presents the time performance statistics of the ASP systems: GRINGO+CLASP, GRINGO+WASP, I-DLV+CLASP and I-DLV+WASP. For example, GRINGO+CLASP indicates the performance of the solver CLASP that is given the ground RABP instance grounded by GRINGO. $u$ is the upper-bound to find a solution, and $c_{max}$ is the computed minimum makespan (i.e. maximum of the activity completion times) by the ASP solver. Figures (1-5) shows the progress of makespan optimization over time until the solver converges to the optimal solution. These figures suggest that the default solving methods of the ASP solvers CLASP and WASP converges at different rates to the optimal solution for the same ground RABP instances.

Table 1: ASP solver time performance of five challenging RABP instances

| | | GRINGO+ CLASP | | GRINGO+ WASP | | I-DLV+ CLASP | | I-DLV+ WASP | |
|---|---|---|---|---|---|---|---|---|---|
| ***id*** | $u$ | *time* | $c_{max}$ | *time* | $c_{max}$ | *time* | $c_{max}$ | *time* | $c_{max}$ |
| **16** | 75 | 935 | *56* | 432 | *56* | 34 | *56* | 56 | *56* |
| **46** | 170 | 812 | *125* | 1186 | *125* | 57 | *125* | 170 | *125* |
| **55** | 380 | 5821 | *163* | 4218 | *163* | 2286 | *163* | 2644 | *163* |
| **59** | 360 | 2032 | *143* | 1305 | *143* | 744 | *143* | 913 | *143* |
| **62** | 150 | 847 | *126* | 1339 | *126* | 266 | *126* | 366 | *126* |

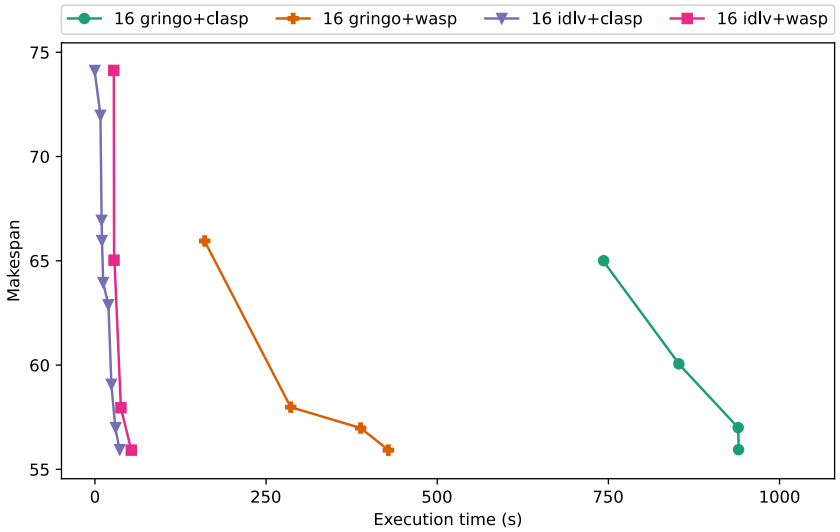



**Fig. 1.** Instance 16

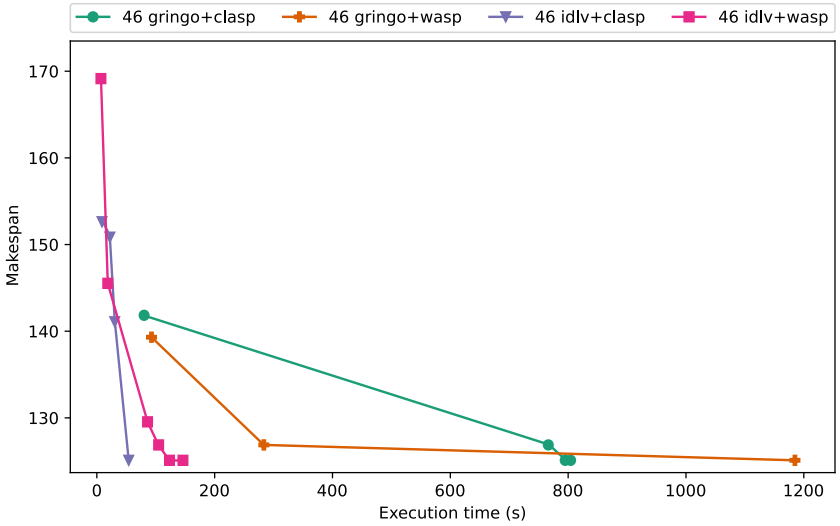



**Fig. 2.** Instance 46

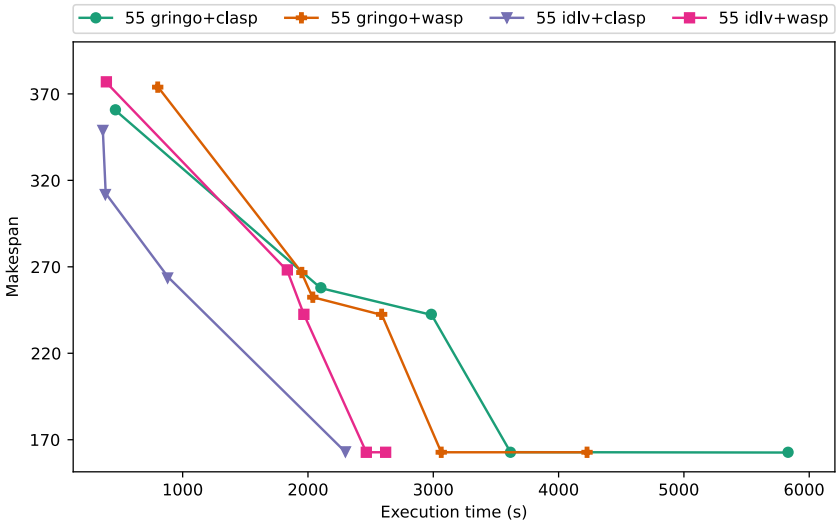



**Fig. 3.** Instance 55

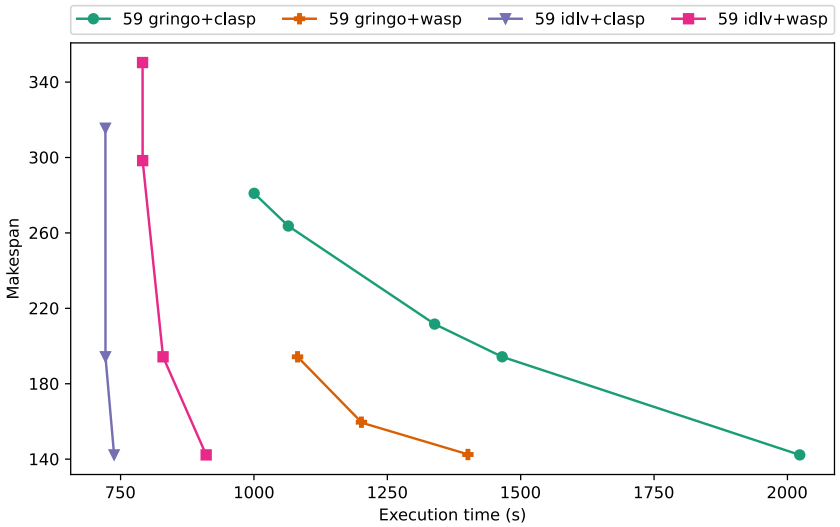



**Fig. 4.** Instance 59

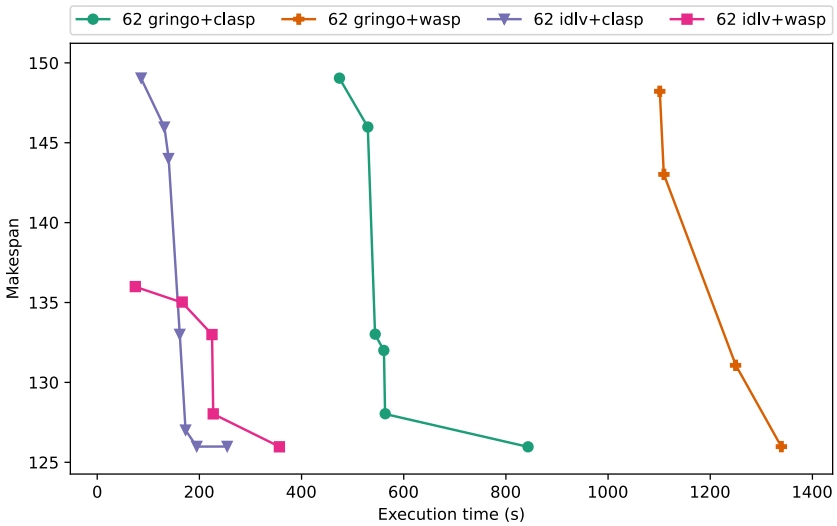



**Fig. 5.** Instance 62

## References


[1] Giray Havur, Cristina Cabanillas, and Axel Polleres. Benchmarking answer set programming systems for resource allocation in business processes. *Expert Systems with Applications*, 205:117599, 2022.

[2] Martin Gebser, Roland Kaminski, Arne König, and Torsten Schaub. Advances in gringo series 3. In James P. Delgrande and Wolfgang Faber, editors, *Proceedings of the 11th International Conference on Logic Programming and Nonmonotonic Reasoning LPNMR 2011*, volume 6645 of *Lecture Notes in Computer Science*, pages 345–351. Springer, 2011.

[3] Francesco Calimeri, Davide Fuscà, Simona Perri, and Jessica Zangari. I-DLV: the new intelligent grounder of DLV. *Intelligenza Artificiale*, 11(1):5–20, 2017.

[4] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Javier Romero, and Torsten Schaub. Progress in clasp series 3. In Francesco Calimeri, Giovambattista Ianni, and Miroslaw Truszczynski, editors, *Proceedings of the 13th International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR 2015*, volume 9345 of *Lecture Notes in Computer Science*, pages 368–383. Springer, 2015.

[5] Mario Alviano, Carmine Dodaro, Nicola Leone, and Francesco Ricca. Advances in WASP. In Francesco Calimeri, Giovambattista Ianni, and Miroslaw Truszczynski, editors, *Proceedings of the 13th International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR 2015*, volume 9345 of *Lecture Notes in Computer Science*, pages 40–54. Springer, 2015.

# Appendix B

# RABP with Makespan Optimization is NP-Hard

**Author:** Giray Havur

In this report, we show that the Resource Allocation in Business Processes [1] (RABP) with makespan minimization is NP-Hard by demonstrating a polynomial reduction from the Job Shop Scheduling [2] (JSS) problem (which is known to be NP-Complete [2, 3]) to RABP problem. We start with defining RABP, and JSS problems.

**Resource Allocation in Business Processes Problem [1]:** A RABP problem is defined by

- Behavioral relations of activities [1]: $\mathcal{B}_{\rightarrow} \subseteq 2^{(A \times A)}$ captures the ordered pairs of activities that are in precedence relation (e.g., $\langle a_1, a_2 \rangle \in B_{\rightarrow}$ represents $a_1 \rightarrow a_2$, which is read as $a_1$ precedes $a_2$), and $\mathcal{B}_{||} \subseteq 2^{(A \times A)}$ represents the ordered pairs of concurrent activities (e.g., $\langle a_1, a_2 \rangle \in \mathcal{B}_{||}$ represents $a_1 || \, a_2$, which is read as $a_1$ is in concurrency relation with $a_2$);
- An organizational model $O = (A, R, L, S_{AL}, S_{RL}, S_{LL})$, where $A$ is a set of *activities*, $R$ is a set of *resources*, $L$ is a set of *roles*, $S_{AL} \subseteq 2^{(A \times L)}$ is a set of activity-to-role assignments specifying which activity can be executed by which role(s), $S_{RL} \subseteq 2^{(R \times L)}$ is the corresponding set of resource-to-role assignment tuples identifying the roles per resource, and optionally, $S_{LL} \subseteq 2^{(L \times L)}$ is a set of role-to-role assignments that form a hierarchical (sub-role) structure;
- A temporal model that contains an upper-bound $u \in U$ on makespan, and resource specific activity durations[1]. The latter is represented as a function $\delta : (R \times A) \rightarrow \mathbb{N}_0$ that maps a resource-activity pair to a duration value in the scope of this report.

A feasible allocation consists of a set of quadruples $I \subseteq 2^{(R \times A \times U \times U)}$ such that $(r_i, a_i, s_i, c_i) \in I$ where each activity $a_i \in A$ is assigned a resource $r_i \in R$, a start time $s_i \in U$, and a completion time $c_i \in U$, $c_i = s_i + \delta(r_i, a_i)$. The following constraints hold for $I$:

1. No activity can be started until the preceding activities in the process model are completed.
   $\forall i_1, i_2 \in I : a_{i_1} \rightarrow a_{i_2} \Rightarrow s_{i_2} \geq c_{i_1}$
2. An activity requires only one resource (that obeys the constraints defined in the Role-Based Access Control [4] (RBAC) model) to be executed.
   $\forall a_i \in A : |\{(r_i, a_i, s_i, c_i) \in I\}| = 1$
   $\exists l \in L \, \forall i_1 \in I : (r_{i_1}, l) \in S_{RL} \wedge (a_{i_1}, l) \in S_{AL}$

[1] This functionality is provided by the *resource-activity duration preference function* in [1, Section 2]

3. A resource can only execute one activity at a time.
   $\forall i_1, i_2 \in I : (s_{i_2} \leq s_{i_1} < c_{i_2}) \Rightarrow r_{i_1} \neq r_{i_2}$
   $\forall i_1, i_2 \in I : (s_{i_2} < c_{i_1} \leq c_{i_2}) \Rightarrow r_{i_1} \neq r_{i_2}$
   $\forall i_1, i_2 \in I : (s_{i_2} > s_{i_1} \wedge c_{i_2} < c_{i_1}) \Rightarrow r_{i_1} \neq r_{i_2}$
4. An activity, once started, must run to completion (i.e., activities are non-preemptive). In other words, no activity can have more than one start time.
   $\forall i_1, i_2 \in I : a_{i_1} = a_{i_2} \Rightarrow s_{i_1} = s_{i_2}$

RABP with makespan optimization aims at finding the feasible allocation(s) with the minimum makespan[2].

**Job Shop Scheduling Problem [2]:** A JSS problem is defined by

- A set of jobs $\mathcal{J} = \{J_1, J_2, \cdots, J_n\}$, where each job $J_i \in \mathcal{J}$ is an ordered set of tasks, i.e., $J_x = \langle t_{x,1}, t_{x,2}, \cdots, t_{x,y} \rangle$;
- A set of machines $M = \{m_1, m_2, \cdots, m_n\}$ for processing tasks;
- A function $\mu : J \rightarrow M$ that maps each task to a machine where it is processed;
- A partial function $PT : (M \times J) \nrightarrow \mathbb{N}_0$ that maps the tasks processed by specific machines to processing times.

A feasible JSS schedule consists of a set of pairs $H \subseteq 2^{(J \times U)}$ such that $(t_{x,y}, b_{x,y}) \in H$ where every task $t_{x,y} \in J_x$ is assigned to a starting time $b_{x,y} \in U$. The processing time required by task $t_{x,y} \in J_x$ processed by machine $m_z$ (i.e., $\mu(t_{x,y})$) is denoted as $p_{x,y,z}$. The completion time $f_{x,y} \in U$ of a task $t_{x,y}$ is the sum of the starting time $b_{x,y}$ and the processing time $p_{x,y,z}$. The following constraints hold for $H$:

1. No task for a job can be started until the previous task for that job is completed.
   $\forall h_{x,y_1}, h_{x,y_2} \in H : y_1 < y_2 \Rightarrow b_{x,y_2} \geq b_{x,y_1} + p_{x,y_1}$
2. A task $t_{x,y}$ requires only one machine to be performed, i.e., the function $\mu(t_{x,y})$ corresponds to the required machine.
3. A machine can only work on one task at a time.
   $\forall h_{x_1,y_1}, h_{x_2,y_2} \in H : (s_{x_2,y_2} \leq s_{x_1,y_1} < f_{x_2,y_2}) \Rightarrow \mu(t_{x_1,y_1}) \neq \mu(t_{x_2,y_2})$
   $\forall h_{x_1,y_1}, h_{x_2,y_2} \in H : (s_{x_2,y_2} < f_{x_1,y_1} \leq f_{x_2,y_2}) \Rightarrow \mu(t_{x_1,y_1}) \neq \mu(t_{x_2,y_2})$
   $\forall h_{x_1,y_1}, h_{x_2,y_2} \in H : (s_{x_2,y_2} > s_{x_1,y_1} \wedge f_{x_2,y_2} < f_{x_1,y_1}) \Rightarrow \mu(t_{x_1,y_1}) \neq \mu(t_{x_2,y_2})$
4. A task, once started, must run to completion (i.e., tasks are non-preemptive).
   $\forall h_{x_1,y_1}, h_{x_2,y_2} \in H : t_{x_1,y_1} = t_{x_2,y_2} \Rightarrow b_{x_1,y_1} = b_{x_2,y_2}$

The objective of the JSS problem is to minimize the completion time of a task (i.e., makespan).

To show a problem $\Pi_1$ is NP-Hard, there needs to be a polynomial reduction from a known NP-complete problem $\Pi_2$, denoted as $\Pi_2 \leq_P \Pi_1$ [5, Chapter 5.1]. We introduce a polynomial reduction algorithm $\mathcal{R}$ in Figure 1 that transforms a JSS problem into a RABP problem. By applying this transformation, the black-box RABP solver $\Pi_{RABP}$ can solve JSS problems (i.e., JSS$\leq_P$RABP).

An overview of the reduction algorithm $\mathcal{R}$ is shown in Figure 2. Given a JSS problem $\{\mathcal{J}, M, \mu, PT\}$, the algorithm computes a RABP problem $\{A, \mathcal{B}, R, L, S_{AL}, S_{RL}, \delta, u\}$ as described in the following.

[2] Makespan is the maximum of completion times (i.e., the completion time of the activity with the latest finish time).

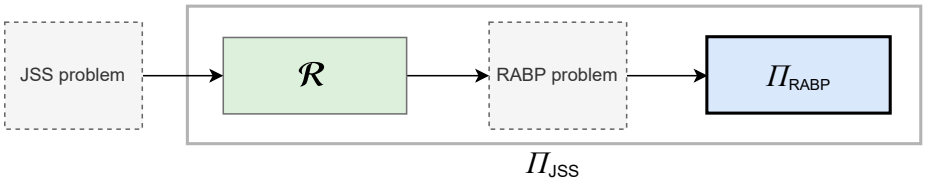



**Fig. 1.** Reduction of a JSS problem to a RABP problem.

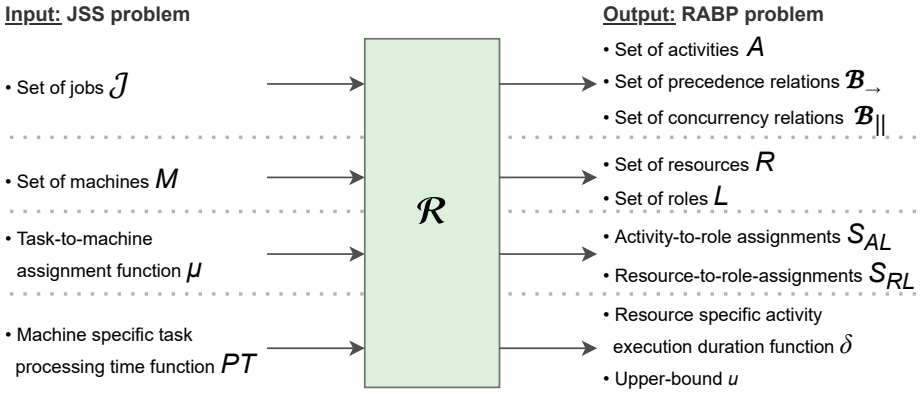



**Fig. 2.** An overview of the polynomial reduction algorithm $\mathcal{R}$.

1. *Set of activities* $A$:
   $A = \{t_{x,y} \mid J_x \in \mathcal{J} \wedge t_{x,y} \in J_x\}$
2. *Set of behavioral relations of activities* $\mathcal{B}_{\rightarrow}$ *and* $\mathcal{B}_{||}$:
   $\mathcal{B}_{\rightarrow} = \{(t_{x,y}, t_{x,y+1}) \mid J_x \in \mathcal{J} \wedge t_{x,y} \in J_x \wedge t_{x,y+1} \in J_x\}$
   $\mathcal{B}_{||} = \{(t_{x,1}, t_{y,1}) \mid J_x \in \mathcal{J} \wedge J_y \in \mathcal{J} \wedge x \neq y \wedge t_{x,1} \in J_x \wedge t_{y,1} \in J_y\}$
3. *Set of resources* $R$:
   $R = \{m_x \mid m_x \in M\}$
4. *Set of roles* $L$:
   $L = \{m_x \mid m_x \in M\}$
5. *Resource-to-role assignments* $S_{RL}$:
   $S_{RL} = \{\langle \mu(t_{x,y}), \mu(t_{x,y})\rangle \mid J_x \in \mathcal{J} \wedge t_{x,y} \in J_x\}$
6. *Activity-to-role assignments* $S_{AL}$:
   $S_{AL} = \{\langle t_{x,y}, \mu(t_{x,y})\rangle \mid J_x \in \mathcal{J} \wedge t_{x,y} \in J_x\}$
7. *Resource specific activity execution duration function* $\delta$:
   $\delta(r, a) = PT(r, a)$
8. *Upper-bound* $u$ *(the worst-case scenario where there is only one machine)*:
   $u = \sum_{J_x \in \mathcal{J}} \sum_{t_{x,y} \in J_x} PT(\mu(t_{x,y}), t_{x,y})$

(1,2) the order of tasks in jobs maps to precedence relations of activities, and the initial tasks of different jobs would be represented as activities in concurrency relation, (3-5)

machines map to roles with only one resource, (6) activity-to-role assignments are derived via task-to-machine assignment function, (7) task durations for specific machines are represented as resource-specific activity durations, and (8) an upper-bound value is given as the sum of all tasks' processing times. Solving the output of $\mathcal{R}$ (e.g. via the Answer Set Programming [6] (ASP) encoding for RABP in [1]) would then yield to a solution to any given JSS problem.

## References


[1] Giray Havur, Cristina Cabanillas, and Axel Polleres. Benchmarking answer set programming systems for resource allocation in business processes. *Expert Systems with Applications*, 205:117599, 2022.

[2] Teofilo Gonzalez and Sartaj Sahni. Flowshop and jobshop schedules: complexity and approximation. *Operations research*, 26(1):36–52, 1978.

[3] Michael R Garey, David S Johnson, and Ravi Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of operations research*, 1(2):117–129, 1976.

[4] Alessandro Colantonio, Roberto Di Pietro, Alberto Ocello, and Nino Vincenzo Verde. A formal framework to elicit roles with business meaning in RBAC systems. In Barbara Carminati and James Joshi, editors, *Proceedings of the 14th ACM Symposium on Access Control Models and Technologies, SACMAT 2009*, pages 85–94. ACM, 2009.

[5] David S. Johnson and Michael R. Garey. Computers and Intractability: A Guide to the Theory of NP-Completeness. *WH Free. Co., San Fr*, 1979.

[6] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

# Appendix C

# A Comparison of ASP and CP Solutions for RABP

**Author:** Giray Havur

**Abstract.** This technical report presents a comparison between declarative implementations of Resource Allocation in Business Processes (RABP) with Answer Set Programming (ASP) and Constraint Programming (CP). We then use the state-of-the-art solvers of these formalisms to solve RABP problem instances of different sizes for reporting on the computational performance of these solvers. The advantages of choosing one formalism over the other are discussed with the following criteria: size of the problem encoding, human readability of the problem instances, and computational performance.

## 1 Introduction

Business process management (BPM) is a field in operations management that focuses on improving corporate performance by managing and optimizing a company's business processes [1]. A process is typically a collection of activities under a control structure serving a particular goal for the business. Resource Allocation in Business Processes (RABP), as an integral part of BPM, deals with the assignment of resources and time intervals to the activities. The complexity of RABP arises from coordinating the dependencies across a broad set of resources and activities of processes as well as from solving potential conflicts on the use of certain resources.

Allocation of resources with starting times to the activities is a far-from-trivial task with strong implications on the quality of the final allocation, and it is challenging from a computational perspective [2]. As for complexity, RABP with makespan optimization is NP-Hard [3](See Appendix B for the proof). One way of addressing this problem is describing it in a knowledge representation formalism. We select ASP [4] and CP [5] for this purpose as they are used to solve various hard computational problems and they maintain a balance between expressivity, ease of use, and computational effectiveness [6, 7]. These two approaches are compared in this particular problem for understanding the respective strengths and weaknesses of these formalisms.

The remainder of this technical report is structured as follows. Section 2 provides the background on the RABP problem, ASP and CP. Based on that, Section 3 presents a comparison of ASP and CP implementations of RABP. Section 4 evaluates the performance of different ASP systems and CP solvers on RABP instances. Section 5 highlights the advantages and limitations different implementations and concludes the paper by giving pointers for future work.

## 2 Background

A background on RABP problem, ASP, and CP is provided in this section.

### 2.1 Resource Allocation in Business Processes Problem

A RABP problem is formalized in [3]. A RABP problem instance is defined by:

- Behavioral relations of activities [3]: $\mathcal{B}_{\rightarrow} \subseteq 2^{(A \times A)}$ captures the ordered pairs of activities that are in precedence relation (e.g., $\langle a_1, a_2 \rangle \in B_{\rightarrow}$ represents $a_1 \rightarrow a_2$, which is read as $a_1$ precedes $a_2$), and $\mathcal{B}_{||} \subseteq 2^{(A \times A)}$ represents the ordered pairs of concurrent activities (e.g., $\langle a_1, a_2 \rangle \in \mathcal{B}_{||}$ represents $a_1 || a_2$, which is read as $a_1$ is in concurrency relation with $a_2$);
- An organizational model $O = (A, R, L, S_{AL}, S_{RL}, S_{LL})$, where $A$ is a set of *activities*, $R$ is a set of *resources*, $L$ is a set of *roles*, $S_{AL} \subseteq 2^{(A \times L)}$ is a set of activity-to-role assignments specifying which activity can be executed by which role(s), $S_{RL} \subseteq 2^{(R \times L)}$ is the corresponding set of resource-to-role assignment tuples identifying the roles per resource, and optionally, $S_{LL} \subseteq 2^{(L \times L)}$ is a set of role-to-role assignments that form a hierarchical (sub-role) structure;
- A temporal model that contains an upper-bound $u \in U$ on makespan[1], and resource specific activity durations[2]. The latter is represented as a function $\delta : (R \times A) \rightarrow \mathbb{N}_0$ that maps a resource-activity pair to a duration value in the scope of this report.

A feasible allocation consists of a set of quadruples $I \subseteq 2^{(R \times A \times U \times U)}$ such that $(r_i, a_i, s_i, c_i) \in I$ where each activity $a_i \in A$ is assigned a resource $r_i \in R$, a start time $s_i \in U$ and a completion time $c_i = s_i + \delta(r, a)$. It is assumed that each activity, once started, is planned to be completed without interruptions in the schedule (i.e. activities are non-preemptive). The following constraints (c.1-c.4) hold for RABP:

**(c.1)** An activity requires only one resource (that obeys the activity-to-role, resource-to-role, and role-to-role assignments defined in the organizational model $O$) to be executed.
$\forall a_i \in A : |\{(r_i, a_i, s_i, c_i) \in I\}| = 1$
$\exists l \in L \; \forall i_1 \in I : (r_{i_1}, l) \in S_{RL} \wedge (a_{i_1}, l) \in S_{AL}$

**(c.2)** Each activity has only one start time.
$\forall i_1, i_2 \in I : a_{i_1} = a_{i_2} \Rightarrow s_{i_1} = s_{i_2}$

**(c.3)** The start time of any activity is greater than or equal to the completion time of its preceding activities.
$\forall i_1, i_2 \in I : a_{i_1} \rightarrow a_{i_2} \Rightarrow s_{i_2} \geq c_{i_1}$

**(c.4)** Same resource must not be allocated to any concurrent pair of activities that have overlapping execution periods.
$\forall i_1, i_2 \in I : (a_{i_1} || a_{i_2} \wedge s_{i_2} \leq s_{i_1} < c_{i_2}) \Rightarrow r_{i_1} \neq r_{i_2}$
$\forall i_1, i_2 \in I : (a_{i_1} || a_{i_2} \wedge s_{i_2} < c_{i_1} \leq c_{i_2}) \Rightarrow r_{i_1} \neq r_{i_2}$
$\forall i_1, i_2 \in I : (a_{i_1} || a_{i_2} \wedge s_{i_2} > s_{i_1} \wedge c_{i_2} < c_{i_1}) \Rightarrow r_{i_1} \neq r_{i_2}$

RABP with makespan optimization aims at finding the feasible allocation(s) with the minimum makespan $u'$ where $0 \leq u' \leq u$.

[1] Makespan is the maximum of completion times (i.e., the completion time of the activity with the latest finish time).

[2] This mapping is further detailed by the *resource-activity duration preference function* in [3, Section 2]

### 2.2 Answer Set Programming (ASP)

An *ASP program* $\Pi$ is a finite set of rules of the form

$$A_0 \text{ :- } A_1, \ldots, A_m, \textit{not } A_{m+1}, \ldots, \textit{not } A_n. \tag{1}$$

where $n \geq m \geq 0$ and each $A_i \in \sigma$ are (function-free first-order) atoms; if $A_0$ is empty in a rule $r$, we call $r$ a constraint, and if $n = m = 0$ we call $r$ a fact.

Whenever $A_i$ is a first-order predicate with variables within a rule of the form (1), this rule is considered as a shortcut for its "grounding" $ground(r)$, i.e., the set of its ground instantiations obtained by replacing the variables with all possible constants occurring in $\Pi$. Likewise, we denote by $ground(\Pi)$ the set of rules obtained from grounding all rules in $\Pi$.

Sets of rules are evaluated in ASP under the so-called stable-model semantics, which allows several models (so called "answer sets"), that is subset-minimal Herbrand models, we again refer to [4] and references therein for details. ASP Solvers typically first compute (a subset of $ground(\Pi)$, and then use a DPLL-like branch and bound algorithm is used to find answer sets for this ground program.

In place of atoms, set-like *choice expressions* of the form $E = \{A_1, \ldots, A_k\}$ can be used. They are true for any subset of $E$; that is, when used in heads of rules, $E$ generates many answer sets, and such rules are often referred to as *choice rules*. Another extension supported in ASP Core-2 standard are optimization statements ([8]) to indicate preferences between possible answer sets:

$$\sim : A_1 : Body_1 = w_1@p_1, \ldots, A_m : Body_m = w_m@p_m$$

associates integer weights (defaulting to 1) with atoms $A_i$ (conditional to $Body_i$ being true), where such a statement expresses that we want to find only answer sets with the smallest aggregated weight sum; again, variables in $A_i : Body_i = w_i@p_1$ are replaced at grounding w.r.t. all possible instantiations.

### 2.3 Constraint Programming (CP)

Constraint programming [5, 9] is a programming paradigm wherein relations between variables are stated in the form of constraints. A Constraint Satisfaction Problem (CSP) consist of a triple $(V, D, C)$ where $V$ is a set of finite domain variables $\{v_1, ..., v_n\}$, $D$ is variable domains $\{dom(v_1), ..., dom(v_n)\}$, and $C$ is a set of constraints defining restrictions on the possible combinations of variable values $(\{c_1, ..., c_m\})$. A solution for a CSP problem is an assignment $S = \{ins(v_1), ..., ins(v_n)\}$ where $ins(v_i) \in dom(v_i)$.

In CSP, in order to make problems easier to solve, *local consistency conditions* [5] are defined. A local consistency condition is a property of constraint satisfaction problems related to the consistency of subsets of variables or constraints. Every local consistency condition can be enforced by a transformation that changes the problem without changing its solutions. Such a transformation is called *constraint propagation* [5]. Constraint propagation works by reducing domains of variables, strengthening constraints, or creating new ones. This leads to a reduction of the search space, making the problem easier to solve by some algorithms. While solving a CSP, *branching strategies* [5]

define the way of partitioning the problem $P$ into easier sub-problems $P_1$, $P_2$,..., $P_n$. To each sub-problem $P_n$, the solver applies again propagation. New branches can be pruned because of the new information derived from the branching. A notable expansion of the traditional CSP model is the *Constrained-Optimization problem* (COP) [5]. A CSP called COP when it has an objective function that has to be optimized (i.e., it can be used for makespan minimization in RABP).

CP problems do not have a common modeling language. A modeling language is usually invented for each CP solver. This makes it challenging for modelers to try out various solvers for problems. A modeling language called MiniZinc [10] is created to facilitate high level modeling and simple experimentation with various solution technologies for the same CP problem. Prior to being presented to solver interfaces, MiniZinc input is converted to FlatZinc [10] constraints that can be consumed by CP solvers. The conversion of MiniZinc to FlatZinc is clearly explained and documented in [10].

## 3 Comparison of ASP and CP Implementations of RABP

In this section, the ASP and CP implementations of RABP are described and compared with respect to human readability and compactness of the problem encodings.

### 3.1 ASP Implementation

The ASP encoding of the RABP problem is provided in Figure 1. Rule (1-3) selects a makespan for the allocation. Rules (4,5) generate the time domain from the selected upper bound. Rule (6) propagates the permissions of activity executions of a junior role to a senior role (i.e. role-to-role RBAC relations). Rules (9-11) implement the resource-activity duration preference handling mechanism described in [3]. Rules (12), (17), (18), and (19-21) correspond to the constraints (c.1), (c.2), (c.3), and (c.4) described in Section 2, respectively. Finally, Rule (22) minimizes the upper bound. Rules (7,8) and (13-16) are added for improving the performance of the ASP encoding by reducing the number of terms in the used predicates.

**Input Format.** The input is divided into three groups of predicates as follows.

*Behavioral relations of activities in a business process $\mathcal{P}$:*

- `activity(`$a$`)`: $a$ is an activity in $\mathcal{P}$;
- `prec(`$a_1$`,`$a_2$`)`: the activity $a_1$ precedes the activity $a_2$ (i.e., $a_1 \rightarrow a_2$);
- `conc(`$a_1$`,`$a_2$`)`: the activity $a_1$ is concurrency with the activity $a_2$ (i.e., $a_1||a_2$).

*RBAC organizational model $O = (A, R, L, S_{AL}, S_{RL}, S_{LL})$:*

- `rlAC(`$r$`,`$l$`)`: resource $r$ has role $l$ (i.e. $(r, l) \in S_{RL}$);
- `alAC(`$a$`,`$l$`)`: the resources with role $l$ can execute activity $a$ (i.e. $(a, l) \in S_{AL}$);
- `llAC(`$l_1$`,`$l_2$`)`: the resources with role $l_1$ can execute the same activities as the resources with role $l_2$ (i.e. $(l_1, l_2) \in S_{LL}$).

*Temporal model:*

- `defActDuration(`$a$`,`$d$`)`: the default duration of activity $a$ is estimated as $d$;
- `rsaDuration(`$r$`,`$a$`,`$d$`)`: the duration of activity $a$ is estimated as $d$ when it is executed by resource $r$;

```
% generation of makespan domain
makespanDomain(U)  :- upperBound(U).                                          1
makespanDomain(U1) :- makespanDomain(U), U1=U-1, U1>=0.                      2

% selection of a makespan
1<={makespan(SU) : makespanDomain(SU)}<=1.                                   3

% time domain generation from the selected makespan
time(0).                                                                     4
time(T1) :- time(T), T1=T+1, T1<=U, makespan(U).                             5

% senior roles can execute activities of junior roles
alAC(A,L1) :- llAC(L1,L2), alAC(A,L2).                                       6

% encoding optimization: projection of duration predicates
prsaDuration(R,A) :- rsaDuration(R,A,X).                                     7
plsaDuration(L,A) :- lsaDuration(L,A,Y).                                     8

% resource-activity duration interval preference
allowedRAD(R,A,D) :- rsaDuration(R,A,D), rlAC(R,L), alAC(A,L).               9
allowedRAD(R,A,D) :- not prsaDuration(R,A), lsaDuration(L,A,D), rlAC(R,L),   10
     alAC(A,L).
allowedRAD(R,A,D) :- not prsaDuration(R,A), not plsaDuration(L,A),           11
     defActDuration(A,D), rlAC(R,L), alAC(A,L).

% (c.1)
1<={allocation(R,A,S,C): time(S), time(C), allowedRAD(R,A,D), C=S+D}<=1 :-   12
     aTransition(A).

% encoding optimization: projection of allocation predicates
pallocation23(A,S) :- allocation(X,A,S,Y).                                   13
pallocation24(A,C) :- allocation(X,A,Y,C).                                   14
pallocation123(R,A,S) :- allocation(R,A,S,X).                                15
pallocation124(R,A,C) :- allocation(R,A,X,C).                                16

% (c.2)
:- pallocation23(A,S1), pallocation23(A,S2), S1<S2.                          17

% (c.3)
:- prec(A1,A2), pallocation24(A1,C1), pallocation23(A2,S2), C1>S2.           18

% (c.4)
:- conc(A1,A2), pallocation123(R,A1,S1), allocation(R,A2,S2,C2), S2<=S1,     19
     C2>S1, A1<A2.
:- conc(A1,A2), pallocation124(R,A1,C1), allocation(R,A2,S2,C2), S2<C1,      20
     C2>=C1, A1<A2.
:- conc(A1,A2), allocation(R,A1,S1,C1), allocation(R,A2,S2,C2), S2>S1,       21
     C2<C1, A1<A2.

% minimization of the selected upper bound
:~ makespan(U). [U,U]                                                        22
```

Fig. 1: ASP encoding for the RABP problem

- `lsaDuration(`$l,a,d$`)`: the duration of activity $a$ is estimated as $d$ when it is executed by a resource that has role $l$;
- `upperBound(`$u$`)`: makespan is bounded at $u$ time units.

**Output Format.** For the allocation output the following predicate is defined:

- `allocation(`$r,a,s,c$`)`: a resource $r$ is allocated to activity $a$ at the start time $s$ until the completion time $c$ (i.e. $(r, a, s, c) \in I$).

```
% input variables
int: nActivities;                                                                    23
int: nResources;                                                                     24
int: upperbound;                                                                     25
int: lastActivity;                                                                   26

set of int: Activity = 1..nActivities;                                               27
set of int: Time = 0..bound;                                                         28
set of int: Resource = 1..nResources;                                                29

% input arrays: preceding activities
array[Activity,Activity] of int: prec;                                               30
%                concurrent activities
array[Activity,Activity] of int: conc;                                               31
%                activity durations
array[Activity,Resource] of int: allowedRAD;                                         32

% output arrays: resource to activity allocations
array[Activity] of var Resource: resource;                                           33
%                 activity to starting time assignments
array[Activity] of var Time: start;                                                  34
%                 activity to ending time assignments
array[Activity] of var Time: end;                                                    35

% (c.1 & c.2)
constraint forall(a in Activity)                                                     36
   (allowedRAD[a,resource[a]]>=0 /\
    aEnd[a]=start[a] + allowedRAD[a,resource[a]]);

% (c.3)
constraint forall(a1 in Activity,a2 in Activity)(                                    37
   if (prec[a1,a2]==1)
   then (start[a2] >= end[a1])
   else true endif);

% (c.4)
constraint forall(a1 in Activity,a2 in Activity)(                                    38
   if (conc[a1,a2]==1 /\
       card(start[a1]..(end[a1]-1) intersect start[a2]..(end[a2]-1))>0)
   then (resource[a1] != resource[a2])
   else true endif);

% makespan minimization
solve minimize (end[lastActivity]);                                                  39
```

Fig. 2: MiniZinc encoding for the RABP problem

### 3.2 MiniZinc Implementation

A possible MiniZinc implementation of RABP is given in Figure 2. Input is defined in Lines (23-32), and output is defined in Lines (33-35). Rules (36), (37), and (38) corresponds to the constraints (c.1 & c.2), (c.3), and (c.4) described in Section 2, respectively. Rule (39) minimizes the makespan.

**Input Format.** The input is divided into three groups of predicates as follows.

*Behavioral relations of activities in a business process $\mathcal{P}$:*

- `nActivities` represents the total number of activities in $\mathcal{P}$,
- `lastActivity` represents the last activity before the end event in $\mathcal{P}$,
- `prec[`$a_1$`,`$a_2$`]` and `conc[`$a_1$`,`$a_2$`]` are 2D arrays representing preceding and concurrency relations between activities: given two activities $a_1$ and $a_2$, if $a_1 \rightarrow a_2$ then

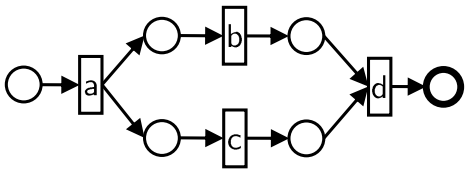



Fig. 3: Example process with four activities represented as a Petri net

| | |
|---|---|
| *Resources* $R$ | res1, res2 |
| *Roles* $L$ | role1, role2 |
| *Activity-to-role assignments* $S_{AL}$ | (a,role1), (b,role2), (c,role1), (d,role2) |
| *Resource-to-role assignments* $S_{RL}$ | (res1,role1), (res2,role2) |
| *Role-to-role assignments* $S_{LL}$ | (role2,role1) |

Fig. 4: Example organizational model

`prec[`$a_1$,$a_2$`]` equals to 1, otherwise 0. Similarly, if $a_1||a_2$ then `conc[`$a_1$,$a_2$`]` equals to 1, otherwise 0.

*Organizational model* $O = (A, R, L, S_{AL}, S_{RL}, S_{LL})$*, and temporal model:*

- `nResources` represents the number of total resources, i.e., $|R|$.
- `upperbound` is the maximum makespan value,
- `allowedRAD[`$a, r$`]` is the 2D array derived from the organizational model $O$ and the temporal constraints: given $O$ and resource-, role-, and default-activity durations [3], if $a$ can be executed by $r$, `allowedRAD[`$a, r$`]`$= d$ where the resource-activity specific duration $d$ is derived from resource-, role-, and default-activity durations.

**Output Format.** The allocated resources, starting, and ending times of an activity $a$ are represented in 1D arrays `resource`, `start`, `end` of length $|A|$ where, e.g., the allocated resource of an activity $a_i$'s is `resource[`$a_i$`]`.

Among these two approaches, RABP problem instances that are represented in ASP are easier to maintain when it becomes necessary to modify the instances due to the predicate structure. The same knowledge is represented in arrays in the RABP problem instances encoded in MiniZinc. On the other hand, the problem encodings (cf. Figure 1 and Figure 2) are of comparible sizes.

## 4 Comparison of A RABP Instance in ASP and CP

In this section, we compare the ASP and CP models of an example RABP instance described in Figures 3, 4, and 5. The process in Figure 3 has four activities: the activity $a$ precedes activities $b$ and $c$, where $b$ and $c$ are concurrent activities. The activities $b$ and $c$ precede the activity $d$. This business process is modeled in ASP by using the predicates `activity` for defining activities, `prec` for determining the precedence of activities, and `conc` for defining the concurrent activities as follows:

| | |
|---|---|
| *Default activity durations* | (a,3), (b,1), (c,2), (d,4) |
| *Role-specific activity durations* | (role2,d,2) |
| *Resource-specific activity durations* | (res1,a,2) |
| *Upper bound* | 15 |

Fig. 5: Example temporal model

```
activity(a). prec(a,b). conc(b,c).
activity(b). prec(a,c). conc(c,b).
activity(c). prec(b,d).
activity(d). prec(c,d).
```

The same process is represented in CP as follows:

```
noActivity = 4;
lastActivity = 4;
prec = [|0,1,1,0,   conc = [|0,0,0,0,
        |0,0,0,1,           |0,0,1,0,
        |0,0,0,1,           |0,1,0,0,
        |0,0,0,0|];         |0,0,0,0|];
```

In the CP process representation, the variable `noActivity` is used to describe the number of activities, the variable `lastActivity` to describe the very last activity in the process which does not precede any other activity, the arrays `prec` and `conc` to describe the precedence and concurrency relations between the activities where rows and columns correspond to activities. For example, in the array `prec`, the value of 1 at the first row (i.e., activity $a$) second column (i.e., activity $b$) means that activity $a$ precedes activity $b$.

The organizational model in Figure 4, and the temporal model in Figure 5 are represented in ASP as follows:

```
resource(res1).  rlAC(res1,role1).  alAC(a,role1).
resource(res2).  rlAC(res2,role2).  alAC(b,role2).
role(role1).                        alAC(c,role1).
role(role2).     llAC(role2,role1). alAC(d,role2).

defaultDuration(a,3). rsaDuration(res1,a,2).
defaultDuration(b,1). lsaDuration(role2,d,2).
defaultDuration(c,2).
defaultDuration(d,4). upperBound(15).
```

By contrast, the CP representation combines the organizational and temporal models in one array named `allowedRAD` where activities index the rows and resources index the columns.

```
noResources = 2;

allowedRAD = [| 2,3,
              |-1,1,
              | 2,2,
              |-1,1|];

upperBound = 15;
```

For example, in the array `allowedRAD`, the value of 2 at the first row (i.e., activity $a$) first column (i.e., resource $res1$) means that $res1$ can be allocated to the activity $a$. If so, execution of activity $a$ by the resource $res1$ takes 2 unit times in the schedule. Any

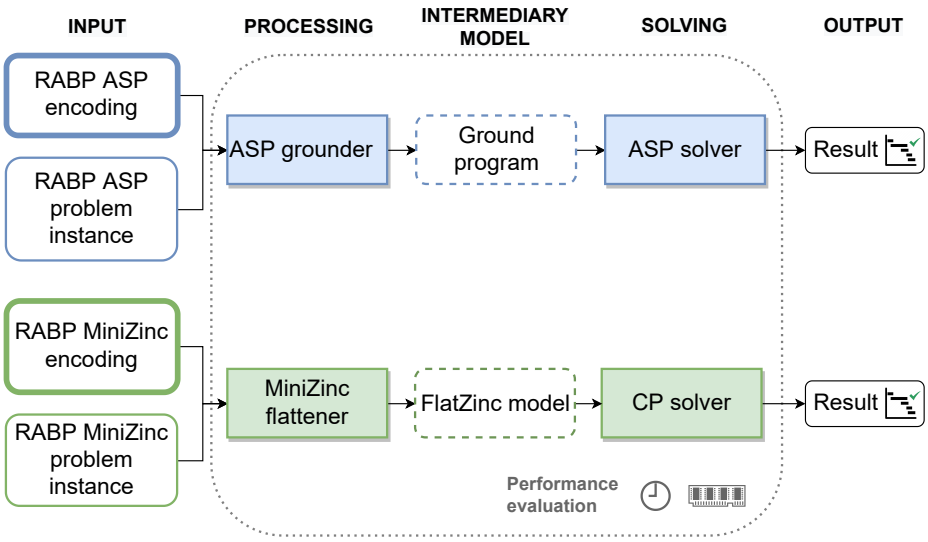



Fig. 6: Overview of ASP and CP solving

negative value in this array means that the corresponding resource cannot be allocated to the corresponding activity.

When ASP and CP RABP instances are compared, ASP language is more suitable for encoding RABP instances from the readability and maintainability perspectives.

## 5 Performance Evaluation

In order to conduct our experiments, we selected four ASP systems and four CP solvers. The ASP systems consist of the combinations of the state-of-the-art ASP grounders GRINGO [11] and I-DLV [12], and the ASP solvers CLASP [13] and WASP [14]. These grounders and solvers are selected for the performance evaluation due to their top performance rankings in the latest ASP Competition [15]. The CP solvers are GECODE [16], CHUFFED [17], HaifaCSP [18], and OR-TOOLS [19] . GECODE and CHUFFED are the out-of-the-box solvers in the MiniZinc environment, and HaifaCSP and OR-TOOLS perform well in the annual MiniZinc challenge[3] [20]. Note that CP solvers take a flattened input: the MiniZinc input is transformed into a low-level FlatZinc representation by MiniZinc-to-FlatZinc converter [10]. Figure 7 summarizes how ASP and MiniZinc inputs (i.e., RABP problem encoding and a problem instance) are transformed into a grounded/flattened (i.e., intermediary) models before being consumed by corresponding solvers.

All the timing results, expressed in seconds, have been obtained by measuring the time needed for computing the optimal RABP solution. Reported use of time and memory includes processing and solving steps (cf. Figure 7). All problem instances have

[3] https://www.minizinc.org/challenge.html

| id | $n_A$ | $\delta_{conc}$ | $n_R$ | $n_L$ | $u$ | id | $n_A$ | $\delta_{conc}$ | $n_R$ | $n_L$ | $u$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 16 | 10 | 2 | 1 | 32 | 22 (2.1) | 16 | 50 | 2 | 1 | 48 |
| 2 | 16 | 50 | 2 | 1 | 32 | 23 (2.2) | 16 | 50 | 2 | 1 | 64 |
| 3 | 16 | 90 | 2 | 1 | 32 | 24 (5.1) | 24 | 50 | 3 | 1 | 72 |
| 4 | 24 | 10 | 3 | 1 | 48 | 25 (5.2) | 24 | 50 | 3 | 1 | 96 |
| 5 | 24 | 50 | 3 | 1 | 48 | 26 (8.1) | 32 | 50 | 4 | 2 | 96 |
| 6 | 24 | 90 | 3 | 1 | 48 | 27 (8.2) | 32 | 50 | 4 | 2 | 128 |
| 7 | 32 | 10 | 4 | 2 | 64 | 28 (11.1) | 48 | 50 | 6 | 3 | 144 |
| 8 | 32 | 50 | 4 | 2 | 64 | 29 (11.2) | 48 | 50 | 6 | 3 | 192 |
| 9 | 32 | 90 | 4 | 2 | 64 | 30 (14.1) | 64 | 50 | 8 | 4 | 192 |
| 10 | 48 | 10 | 6 | 3 | 96 | 31 (14.2) | 64 | 50 | 8 | 4 | 256 |
| 11 | 48 | 50 | 6 | 3 | 96 | 32 (17.1) | 96 | 50 | 12 | 6 | 288 |
| 12 | 48 | 90 | 6 | 3 | 96 | 33 (17.2) | 96 | 50 | 12 | 6 | 384 |
| 13 | 64 | 10 | 8 | 4 | 128 | 34 (20.1) | 128 | 50 | 16 | 8 | 384 |
| 14 | 64 | 50 | 8 | 4 | 128 | 35 (20.2) | 128 | 50 | 16 | 8 | 512 |
| 15 | 64 | 90 | 8 | 4 | 128 | | | | | | |
| 16 | 96 | 10 | 12 | 6 | 192 | | | | | | |
| 17 | 96 | 50 | 12 | 6 | 192 | | | | | | |
| 18 | 96 | 90 | 12 | 6 | 192 | | | | | | |
| 19 | 128 | 10 | 16 | 8 | 256 | | | | | | |
| 20 | 128 | 50 | 16 | 8 | 256 | | | | | | |
| 21 | 128 | 90 | 16 | 8 | 256 | | | | | | |

Table 1: Properties of problem instances

been solved on a PC (Intel Xeon processor 2.8 GHz) running a Linux kernel. Time and memory for each run were limited to 1 h CPU clock time and 16 GB of memory usage, respectively. The RABP encodings, and the problem instances are accessible at `http://urban.ai.wu.ac.at/~havur/RABPASPCPComparison`.

**RABP Instances.** We use BRANCH [3, 21] for generating 35 RABP problem instances. The details of these instances are provided in Table 1. In the table, *id* column is the unique identifier of each instance, $n_A$ is the number of activities to which resources are going to be allocated, $\delta_{conc}$ is the degree of concurrency of the generated Petri net, $n_R$ is the number of resources, $n_L$ is the number of roles, and $u$ is the upper bound for makespan. Problem instances (1-21) have gradually increasing number of activities and resources, and problem instances (22-35) are repeated instances of (2,5,8,11,14,17,20) where the upper bound values are doubled and tripled. With the latter, we test the solvers for makespan optimization (i.e., with higher initial upper bounds).

**Results.** The experiment results are shown in Table 2. It is clear that the memory usage of the CP solvers are consistently lower than the ASP solvers. These results are visualized in Figure 7 as a scatter plot where the x-axis represents the time and the y-axis represents the memory usage of the problem instances solved by the respective systems that are indicated by different markers.

The sorted cactus plots in Figure 8 visualizes the total number of successfully-solved problem instances with respect to required time to solve these instances. These two figures suggest that both ASP systems and CP solvers are competitive in solving RABP instances, while the CP solver HaifaCSP outperforms the others.

| | ASP | | | | | | | | CP | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | gringo+clasp | | gringo+wasp | | idlv+clasp | | idlv+wasp | | gecode | | chuffed | | hcsp | | or-tools | |
| **id** | time | mem | time | mem | time | mem | time | mem | time | mem | time | mem | time | mem | time | mem |
| 1 | 0.79 | 13.3 | 0.93 | 16 | 0.79 | 12.5 | 0.98 | 16.8 | 0.22 | 26.1 | 0.47 | 35.8 | 0.59 | 45 | 0.58 | 39.2 |
| 2 | 9.13 | 14.5 | 34.77 | 136.8 | 27.58 | 15.8 | 20.42 | 99.5 | 469.68 | 27.8 | 9.67 | 77.1 | 1.06 | 58.6 | 717.9 | 55.6 |
| 3 | **OoT** | 51.2 | **OoT** | 1052.7 | **OoT** | 43 | **OoT** | 1018.6 | **OoT** | 27.9 | 2872.33 | 653.2 | 1.28 | 62.3 | **OoT** | 57.5 |
| 4 | 2.84 | 24.1 | 3.24 | 41.6 | 2.73 | 22.6 | 10.3 | 39.2 | 0.22 | 22.2 | 0.83 | 59.7 | 1.47 | 76.3 | 1.26 | 67.1 |
| 5 | 4.29 | 29.8 | 5.04 | 53.1 | 3.5 | 24.3 | 11.67 | 50.4 | 4.57 | 33.9 | 1.91 | 105.1 | 3.48 | 135.1 | **OoT** | 102.8 |
| 6 | **OoT** | 121.3 | **OoT** | 2010.5 | **OoT** | 112.5 | **OoT** | 1403.3 | **OoT** | 37.4 | **OoT** | 762.7 | 4.38 | 154.8 | **OoT** | 115.9 |
| 7 | 6.85 | 39.3 | 8.32 | 79.5 | 5.97 | 38 | 43.95 | 79.3 | 0.34 | 37.2 | 1.72 | 102.4 | 2.84 | 132.6 | 2.19 | 100.9 |
| 8 | 10.97 | 70 | 15.47 | 141.7 | 9.73 | 48.2 | 75.57 | 111.8 | 53.83 | 44.1 | 4.9 | 211.8 | 10.28 | 279.7 | **OoT** | 192.4 |
| 9 | **OoT** | 154.1 | **OoT** | 4718.1 | **OoT** | 150.2 | **OoT** | 2712.3 | **OoT** | 47.7 | **OoT** | 911.5 | 13.39 | 336.8 | **OoT** | 223.8 |
| 10 | 45.25 | 132.3 | 39.39 | 231.5 | 45.65 | 113.6 | 1023.78 | 293.6 | 0.57 | 54.1 | 5.08 | 233.9 | 10.88 | 310 | 8.24 | 215.3 |
| 11 | 56.69 | 271 | 53.51 | 233.8 | 81.99 | 149.3 | 104.79 | 318.9 | 284.51 | 73.3 | 23.52 | 640.2 | 57.03 | 869.8 | **OoT** | 554.4 |
| 12 | **OoT** | 290.2 | **OoT** | 8771 | **OoT** | 231.2 | **OoT** | 5610.8 | **OoT** | 85.8 | **OoT** | 1361.3 | 77.97 | 1062.9 | **OoT** | 679.2 |
| 13 | 464.52 | 363 | 812.21 | 1527.1 | 289.59 | 289.8 | 726.64 | 1415.3 | 0.99 | 76.8 | 10.89 | 416.9 | 22.72 | 559.2 | 22.5 | 373.9 |
| 14 | 209.68 | 756.7 | 221.86 | 417.9 | 570.92 | 373.5 | 647.89 | 796.5 | **OoT** | 113 | 72.23 | 1432.6 | 208.73 | 1954.9 | **OoT** | 1224.5 |
| 15 | **OoT** | 964.7 | **OoT** | 10038.5 | **OoT** | 415.2 | **OoT** | 11827.8 | **OoT** | 139.4 | **OoT** | 2314.1 | 320.6 | 2472.7 | **OoT** | 1530.8 |
| 16 | **OoT** | 1321.9 | **OoT** | 5580.8 | **OoT** | 1047.8 | **OoT** | 6050.2 | 2.23 | 145.5 | 42.44 | 1166.9 | 121.56 | 1592.1 | 161.7 | 1011.5 |
| 17 | 1489.66 | 3620.8 | 1334.9 | 1340.8 | 2857.3 | 1459.2 | 2974.85 | 3107.9 | **OoT** | 229.1 | 462.88 | 4806.9 | 1395.15 | 6422.7 | 2504.33 | 3979.6 |
| 18 | **OoT** | 4294.3 | **OoT** | 15415 | **OoT** | 257.3 | **OoT** | 1578 | **OoT** | 253.5 | **OoT** | 6512.9 | 2099.15 | 8400.8 | **OoT** | 5198.2 |
| 19 | **OoT** | 3334.5 | **OoT** | 7747 | **OoT** | 612.9 | **OoT** | 3723.2 | 4.24 | 254.7 | 75.39 | 2386 | 349.2 | 3204.6 | 508.23 | 2036 |
| 20 | **OoT** | 10420.1 | **OoT** | 6398.1 | **OoT** | 169 | **OoT** | 168.9 | **OoT** | 405.6 | 1613.37 | 11024.1 | **OoT** | 14700.9 | **OoT** | 9063.2 |
| 21 | **OoT** | 13271.9 | **OoT** | 4831.9 | **OoT** | 170.7 | **OoT** | 170.8 | **OoT** | 467.9 | **OoT** | 14825.8 | 338.29 | **OoM** | **OoT** | 11940 |
| 22 | 11.69 | 18.9 | 29.03 | 144.4 | 14.96 | 16.7 | 32.88 | 139 | 430.19 | 29.3 | 7.11 | 81.1 | 1.29 | 74.5 | 1504.59 | 65.4 |
| 23 | 13.06 | 23.5 | 70.74 | 289.3 | 25.13 | 21.6 | 67.17 | 250 | 431.93 | 30.6 | 9.8 | 99.1 | 2.17 | 89.8 | **OoT** | 75.5 |
| 24 | 6.4 | 44.9 | 9.1 | 82.7 | 6.29 | 37.8 | 44.36 | 80.8 | 2.4 | 37.9 | 3.32 | 141.9 | 6.06 | 187.3 | **OoT** | 133.4 |
| 25 | 10.73 | 66.8 | 13.84 | 118.2 | 9.41 | 53.6 | 130.93 | 115.8 | 3.09 | 41.1 | 5.58 | 182.7 | 8.17 | 239.9 | **OoT** | 168.5 |
| 26 | 21.42 | 123.3 | 28.12 | 223.2 | 21.01 | 80.1 | 365.73 | 185 | 70.33 | 51.5 | 10.36 | 304.2 | 18.87 | 408.8 | **OoT** | 271.2 |
| 27 | 33.44 | 176 | 32.38 | 191.5 | 40.43 | 115.6 | 1202.17 | 273.5 | 58.26 | 58 | 15.8 | 397.2 | 24.9 | 527.8 | **OoT** | 346.6 |
| 28 | 101.12 | 460.8 | 114.47 | 293.4 | 248.52 | 253.3 | 265.02 | 463.2 | 304.19 | 90 | 49.47 | 929.6 | 105.35 | 1266.1 | **OoT** | 794.3 |
| 29 | 186.69 | 650.9 | 182.1 | 402 | 412.67 | 411 | 443.47 | 639.7 | 310.28 | 103.7 | 107.67 | 1246.5 | 160.08 | 1687.7 | **OoT** | 1056.3 |
| 30 | 396.94 | 1267.3 | 383.27 | 642.2 | 1345.16 | 669.9 | 1440.37 | 1188.2 | **OoT** | 144.1 | 193.9 | 2163.7 | 409.06 | 2912.6 | **OoT** | 1826.2 |
| 31 | 588.57 | 1897 | 646.59 | 819.2 | 2310.91 | 1017.7 | 2391.43 | 1550.3 | **OoT** | 166.5 | 248.49 | 2918.7 | 595.93 | 3822.2 | **OoT** | 2383.8 |
| 32 | 2975.27 | 5633.7 | 2580.04 | 2060.3 | **OoT** | 105.2 | **OoT** | 105.1 | **OoT** | 298.4 | 847.71 | 7087.6 | 2723.18 | 9525.1 | 3192.26 | 5832.6 |
| 33 | **OoT** | 8298.5 | 3374.496 | 3133.4 | **OoT** | 142.3 | **OoT** | 142.3 | **OoT** | 358.4 | 1866.95 | 9870.6 | **OoT** | 12631.3 | **OoT** | 7732.9 |
| 34 | **OoT** | 336.3 | **OoT** | 326.3 | 1266.3 | **OoM** | 405.6 | **OoM** | **OoT** | 523 | 334.58 | **OoM** | 347.63 | **OoM** | **OoT** | 13357.1 |
| 35 | **OoT** | 202.2 | **OoT** | 202.3 | **OoT** | 327.8 | **OoT** | 327.8 | **OoT** | 633.4 | 349.1 | **OoM** | 423.59 | **OoM** | 467.3 | **OoM** |

Table 2: Evaluation results

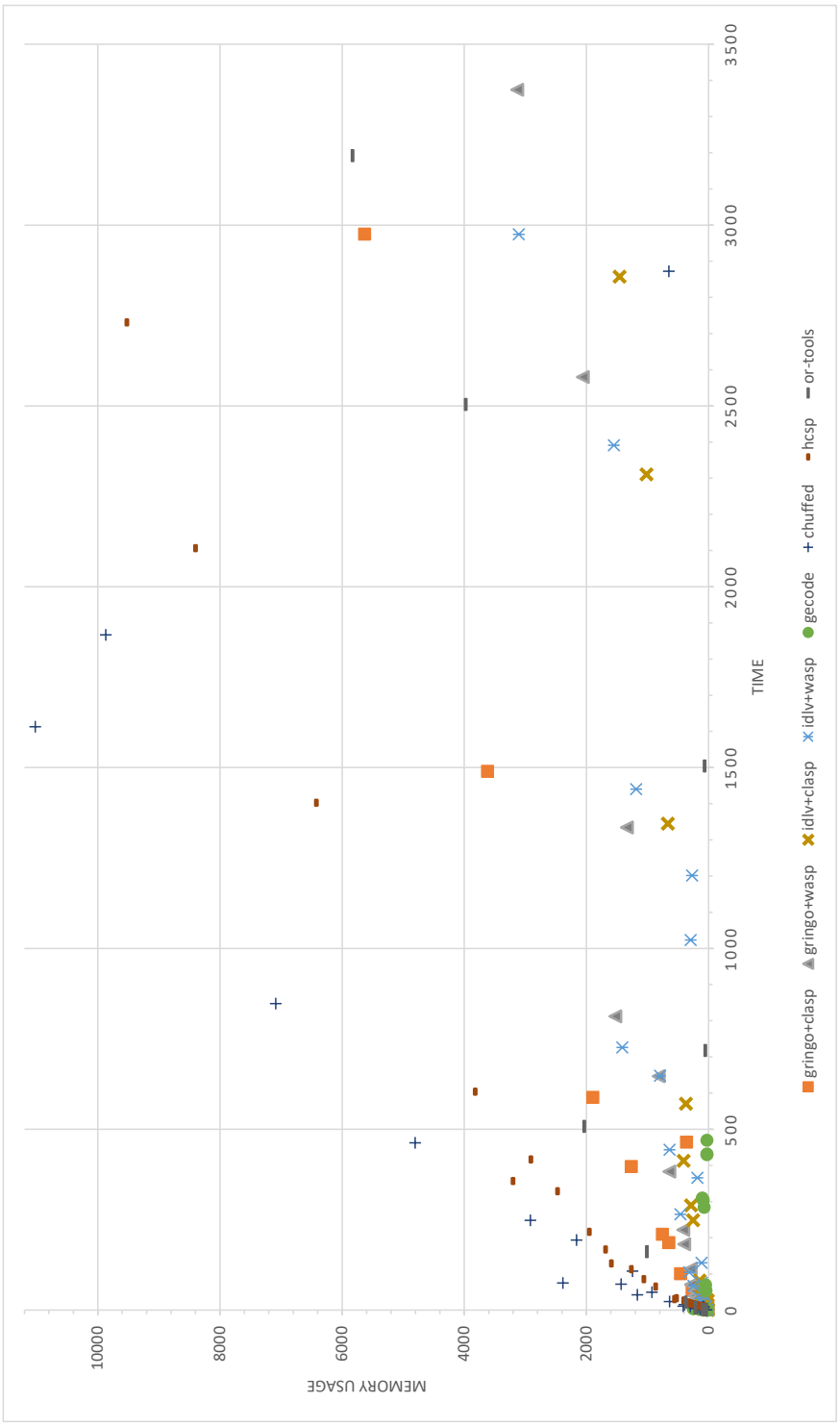



Fig. 7: Scatter plot of all instances

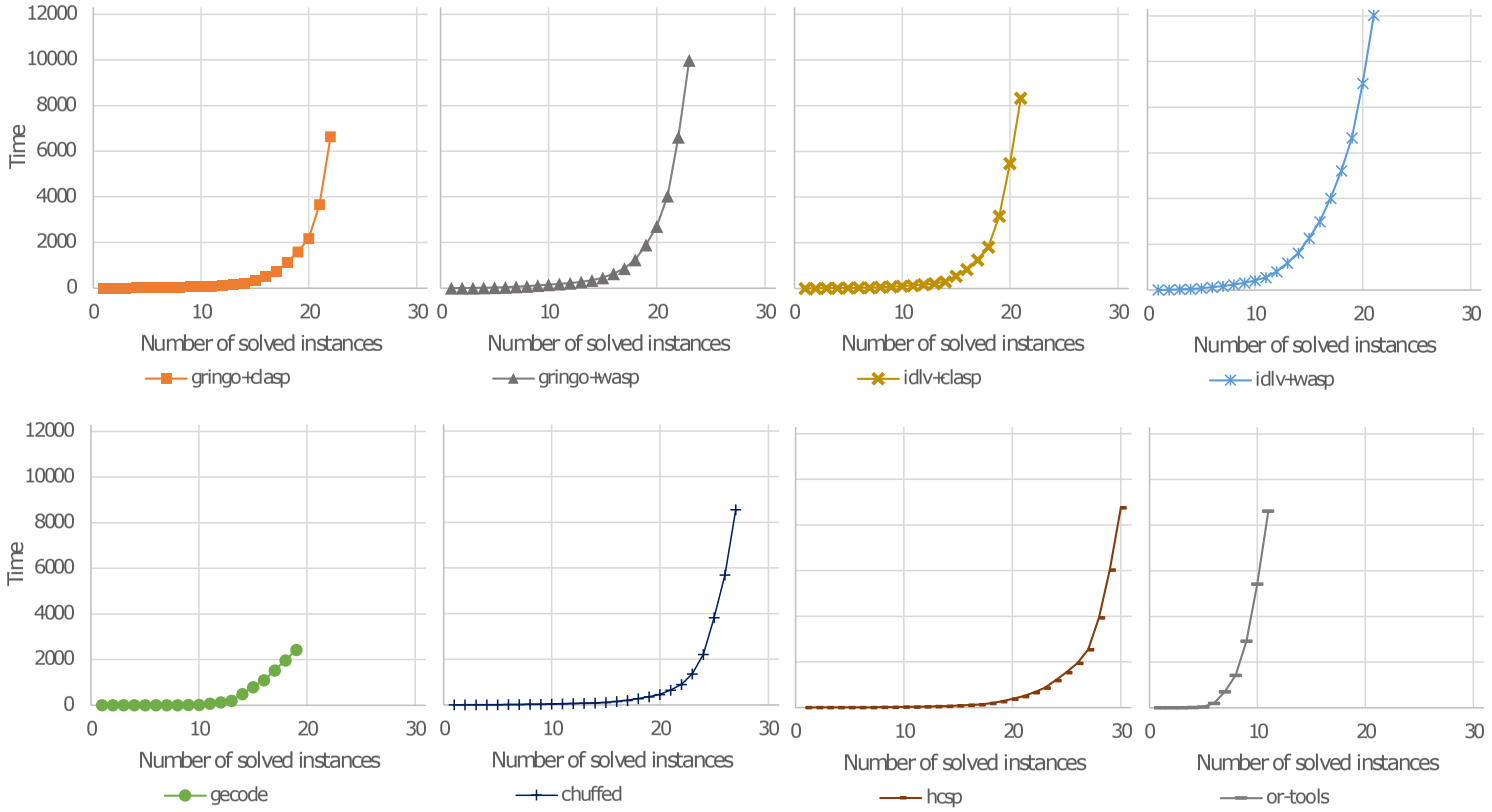



Fig. 8: Sorted cactus plots

Overall, the ASP systems GRINGO+CLASP, GRINGO+WASP, IDLV+CLASP, and IDLV+WASP solve 22, 35, 23, and 21 problem instances; whereas the CP solvers GECODE, CHUFFED, HCSP, and OR-TOOLS solve 19, 27, 30, and 11 problem instances among the given 35 instances whose properties are detailed in Table 1.

## References


[1] Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A. Reijers. *Fundamentals of Business Process Management*. Springer, 2 edition, 2018.

[2] Michele Lombardi and Michela Milano. Optimal methods for resource allocation and scheduling: a cross-disciplinary survey. *Constraints*, 17(1):51–85, 2012.

[3] Giray Havur, Cristina Cabanillas, and Axel Polleres. Benchmarking answer set programming systems for resource allocation in business processes. *Expert Systems with Applications*, 205:117599, 2022.

[4] Gerhard Brewka, Thomas Eiter, and Mirosław Truszczyński. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2011.

[5] Francesca Rossi, Peter Van Beek, and Toby Walsh. *Handbook of constraint programming*. Elsevier, 2006.

[6] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

[7] Antoni Niederliński. *A Quick and Gentle Guide to Constraint Logic Programming Via ECLiPSe*. Jacek Skalmierski Computer Studio, 2012.

[8] Francesco Calimeri, Wolfgang Faber, Martin Gebser, Giovambattista Ianni, Roland Kaminski, Thomas Krennwallner, Nicola Leone, Marco Maratea, Francesco Ricca, and Torsten Schaub. ASP-Core-2 input language format. *Theory and Practice of Logic Programming*, 20(2):294–309, 2020.

[9] Kim Marriott and Peter J Stuckey. *Programming with constraints: an introduction*. MIT press, 1998.

[10] Nicholas Nethercote, Peter J Stuckey, Ralph Becket, Sebastian Brand, Gregory J Duck, and Guido Tack. Minizinc: Towards a standard CP modelling language. In *International Conference on Principles and Practice of Constraint Programming*, pages 529–543. Springer, 2007.

[11] Martin Gebser, Roland Kaminski, Arne König, and Torsten Schaub. Advances in gringo series 3. In James P. Delgrande and Wolfgang Faber, editors, *Proceedings of the 11th International Conference on Logic Programming and Nonmonotonic Reasoning LPNMR 2011*, volume 6645 of *Lecture Notes in Computer Science*, pages 345–351. Springer, 2011.

[12] Francesco Calimeri, Davide Fuscà, Simona Perri, and Jessica Zangari. I-DLV: the new intelligent grounder of DLV. *Intelligenza Artificiale*, 11(1):5–20, 2017.

[13] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Javier Romero, and Torsten Schaub. Progress in clasp series 3. In Francesco Calimeri, Giovambattista Ianni, and Miroslaw Truszczynski, editors, *Proceedings of the 13th International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR 2015*, volume 9345 of *Lecture Notes in Computer Science*, pages 368–383. Springer, 2015.

[14] Mario Alviano, Francesco Calimeri, Carmine Dodaro, Davide Fuscà, Nicola Leone, Simona Perri, Francesco Ricca, Pierfrancesco Veltri, and Jessica Zangari. The ASP system DLV2. In Marcello Balduccini and Tomi Janhunen, editors, *Proceedings of the 14th International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR 2017*, volume 10377 of *Lecture Notes in Computer Science*, pages 215–221. Springer, 2017.

[15] Martin Gebser, Marco Maratea, and Francesco Ricca. The seventh answer set programming competition: Design and results. *Theory and Practice of Logic Programming*, 20(2):176–204, 2020.

[16] Christian Schulte, Mikael Lagerkvist, and Guido Tack. Gecode. *Software download and online material at the website: http://www. gecode. org*, pages 11–13, 2006.

[17] Geoffrey Chu, Peter J Stuckey, Andreas Schutt, Thorsten Ehlers, Graeme Gange, and Kathryn Francis. Chuffed: A lazy clause solver, 2010.

[18] Michael Veksler and Ofer Strichman. A proof-producing csp solver. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[19] Google. Google Optimization Tools. W3C Recommendation, March 2017. `https://developers.google.com/optimization/`.

[20] Peter J Stuckey, Thibaut Feydy, Andreas Schutt, Guido Tack, and Julien Fischer. The minizinc challenge 2008–2013. *AI Magazine*, 35(2):55–60, 2014.

[21] Giray Havur, Cristina Cabanillas Macías, and Axel Polleres. Branch: an asp systems benchmark for resource allocation in business processes. In *BPM 2021: Best Dissertation Award, Doctoral Consortium, and Demonstration & Resources Track co-located with the 19th International Conference on Business Process Management (2021), pp. 176-180.* CEUR Workshop Proceedings (CEUR-WS. org), 2021.