



HAL
open science

Cost optimization of business processes based on time constraints on cloud resources

Rania Ben Halima

► **To cite this version:**

Rania Ben Halima. Cost optimization of business processes based on time constraints on cloud resources. Networking and Internet Architecture [cs.NI]. Institut Polytechnique de Paris, 2020. English. NNT : 2020IPPAS014 . tel-02969494

HAL Id: tel-02969494

<https://theses.hal.science/tel-02969494v1>

Submitted on 16 Oct 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



INSTITUT
POLYTECHNIQUE
DE PARIS

NNT : 2020IPPAS014

Thèse de doctorat



Optimisation du coût des processus métier basée sur des contraintes temporelles sur ressources Cloud

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom SudParis

École doctorale n°626 :
Ecole Doctorale de l'Institut Polytechnique de Paris (ED IP Paris)



INSTITUT
POLYTECHNIQUE
DE PARIS

Spécialité de doctorat: Informatique

Thèse présentée et soutenue à Evry, le 15/09/2020, par

Rania Ben Halima Kchaou

Composition du Jury :

Karim Baïna Professeur, Université Mohammed V de Rabat, Maroc	Président
Aly Megahed Research Manager, IBM's Almaden Research Center, United States	Rapporteur
Narjès Bellamine Ben Saoud Professeur, Université de la Manouba, Tunisie	Rapporteur
Mohamed Jmaiel Professeur, Université de Sfax, Tunisie	Examineur
Walid Gaaloul Professeur, TELECOM SudParis, France	Directeur de thèse
Slim Kallel Maitre-Assistant, Université de Sfax, Tunisie	Co-encadrant de thèse

Titre : Optimisation du coût des processus métier basée sur des contraintes temporelles sur ressources cloud

Mots clés : Vérification, Optimisation, Processus métier, Ressource cloud

Résumé : Motivé par le besoin "d'optimiser le coût de déploiement des processus métier" les organisations externalisent certaines de leurs opérations de processus métiers vers le cloud computing. Les fournisseurs cloud proposent des stratégies de tarification compétitives (par exemple, à la demande, réservée, ponctuelle) spécifiées en fonction des contraintes temporelles pour répondre aux demandes de changement et de dernières minutes. En outre, les processus métier ont des contraintes temporelles et toute violation de ces contraintes pourrait entraîner des conséquences sévères. Par conséquent, il est nécessaire de vérifier formellement que l'allocation des ressources cloud dans le processus métier est temporellement correcte. Cependant, en l'absence d'une définition formelle des stratégies de tarification cloud, spécifiée en langage naturel, la consistance temporelle de l'allocation des ressources cloud ne peut pas être vérifiée dans le contexte de gestion de processus métier. En outre, la variété des ressources cloud, des

stratégies de tarification et des exigences des activités ne permettent pas au concepteur du processus métier de trouver facilement le coût de déploiement optimal pour un processus métier. Dans cette thèse, nous visons à: (i) améliorer le support des contraintes temporelles des activités et les disponibilités temporelles des ressources cloud ainsi que les stratégies de tarification, (ii) fournir deux solutions pour minimiser le coût de déploiement des processus métier. Pour ce faire, nous proposons une spécification formelle des ressources cloud, des stratégies de prix et des contraintes temporelles des activités. Cette spécification est utilisée pour vérifier formellement la consistance temporelle de l'allocation des ressources cloud dans un processus métier enrichi par des contraintes temporelles. Ensuite, nous proposons deux modèles de programmation linéaire, program linéaire binaire et program à entier mixte, pour trouver le coût optimal de déploiement d'un processus métier dans les ressources cloud.

Title : Cost optimization of business processes based on time constraints on Cloud resources

Keywords : Verification, Optimization, Business process, Cloud resource

Abstract : Motivated by the need of "optimizing the deployment cost of business processes" organizations outsource some of their operations to cloud computing. Cloud providers offer competitive pricing strategies (e.g., on-demand, reserved, and spot) specified based on temporal constraints to accommodate users' changing and last-minute demands. Besides, the organizations' business processes are time constrained and any violation to these constraints could lead to serious consequences. Therefore, there is a need to formally verify that the cloud resource allocation in a business process is temporally correct. However, due to the lack of a formal definition of cloud pricing strategies, specified in natural language, the temporal correctness of cloud resource allocation in a business process management context can not be verified. Furthermore, the

variety of cloud resources, pricing strategies, and activities requirements do not help the business process designer to easily find the optimal business process's deployment cost. In this thesis, our objectives are to: (i) improve the business processes support of temporal constraints on activities and cloud resources, as well as pricing strategies and (ii) minimize the business process deployment cost. To this end, we propose a formal specification for cloud resources, pricing strategies, and activities' temporal constraints. This specification is used to formally verify the temporal correctness of cloud resource allocation in time-aware business processes. Then, we propose two linear program models, binary linear program and mixed integer program, to find the optimal deployment cost of time-aware business processes in cloud resources.

Acknowledgment

First and foremost, I thank God for giving me strength, knowledge, ability and opportunity to undertake this research work and to continue and complete it satisfactorily. Without his blessings, this achievement would not have been possible.

I would like to thank all the jury members. I thank Professor Narjès Bellamine Ben Saoud and Dr Aly Megahed for accepting being my thesis reviewers and for their attention and thoughtful comments. I also thank Professor Amel Bouzeghoub, Professor Mohamed Jmaiel, and Professor Karim Baïna for accepting being my thesis examiners.

I would like to express my appreciation and gratitude to my supervisor Walid Gaaloul. His valuable advice, patience, enthusiasm and constant support all the time of research allowed me to acquire new understandings and extend my experiences. He was not only an advisor but also a good friend. Thank you for your guidance, it has been a true pleasure and I deeply hope that we can continue our collaboration.

A special gratitude is also due to my co-supervisor Slim Kallel for his great advices and his guidance. I am thankful for the opportunities he provided, and for having faith in me. I am deeply grateful for the great deal of time we spent discussing many technical details of our work together. Special thanks to Dr Kais Klai, Professor Zakaria Maamar, Dr Mehdi Ahmed-Nacer, and Mme Imen Zouaghi for the beneficial collaborations we have had.

I am indebted grateful to Professor Carlos Juiz Garcìa for his feedbacks and suggestions on my work during my mobility in the University of the Balearic Islands.

I owe my deepest gratitude and warmest affection to the members of the computer science department of Telecom SudParis. I would like to thank Brigitte Houassine for her kind help and assistance. A special thank you to my office mates Emna, Souha, Rami, Hayet, Kunal, Nabila, Ikram, Aicha, and Leila.

A special thank to the RedCAD laboratory members Riadh, Saoussen, Fairouz, and Feten for all their help and support.

Thanks to my friends especially Ghassen, Maroi, Nada, Hiba, and Haythem for their emotional support and for all the beautiful moments we shared in France and Tunisia.

I am forever thankful to my parents: my father Abdellatif, my mother Soumaya, and my brothers Yessine and Mahdi who were always there for me with encouraging words whenever I started doubting myself. Your encouragement made me go forward and made me want to succeed. My warm thanks to my parents-in-law, Taoufik and Sajia for their love and kindness. I am also grateful to my brother in law Walid, my sister in law Feten, and their cute children Iyed and Ayoub, my brother in law Wassim, my sister in law Lilia, and their sweet children Yessmine, Edam, and Mohamed, my brother in law Amine, my sister in law Molka, and their lovely son Ismaiel. Thank you so much for having faith in me! You supported me without even you know it.

I express my deepest gratitude to my soul mate and my loving husband Ilyes. His

love and encouragement fostered me to concentrate at work. His understanding and support helped me to get through many difficult times. I hope that my thesis will be a source of pride for you.

Finally, I dedicate this thesis to my little baby still fetus. I hope that you will be proud of your mum.

I love you so much.

Rania Ben Halima Kchaou

Abstract

Organizations are recently more and more adopting Process-Aware Information Systems for the management and the execution of their business processes. Motivated by a high-level performance at deployment and execution while keeping reduced development and maintenance costs, organizations have started outsourcing their business processes using cloud computing resources. Cloud is an increasingly popular computing paradigm that provides on-demand services over the Internet as it reduces organizations' needs to plan ahead for provisioning resources. Many cloud providers offer competitive pricing strategies (e.g., on-demand, reserved, and spot) to accommodate users' changing and last-minute demands.

The design of correct time-aware business processes in cloud resources with respect to temporal constraints has become challenging. Since business processes are time-constrained and pricing strategies are specified based on temporal constraints, modeling correct process models is undoubtedly a difficult and an error-prone task. The resource perspective in business process models is poorly operated in comparison to other perspectives such as the control-flow. Although several approaches have been proposed in the literature, they all targeted cloud resources properties such as elasticity and shareability rather than pricing strategies, i.e., temporal availabilities.

The optimization of the deployment cost of time-aware business processes in different cloud resources proposed under various pricing strategies becomes a highly challenging problem. Despite their varieties and benefits to optimize deployment cost, using pricing strategies can lead to exceeding budget constraints due to inappropriate decisions when allocating cloud resources to business processes. Different proposals have been suggested in the literature to optimize the deployment cost of business processes. They, however, consider neither the variety of pricing strategies nor the advanced temporal constraints.

In this thesis, we address the above shortcomings by proposing an approach for optimizing the deployment cost of business processes in cloud resources based on temporal constraints. We aim to: (i) *improve the business processes support of temporal constraints on activities and cloud resources, as well as pricing strategies* and (ii) *provide solutions to minimize the business process deployment cost*. To this end, we propose a formal specification for cloud resources, pricing strategies, and activities' temporal constraints. This specification is used to formally verify the temporal correctness of cloud resource allocation in time-aware business processes at design time. Then, we propose two linear program models, binary linear program and mixed integer program, to find the optimal deployment cost of time-aware business processes in cloud resources. To validate our approach, we (i) develop a proof of concepts as an extension for existing business process modeling tools, and (ii) perform experiments to validate our approach. Experiments have proven the effectiveness, feasibility, and scalability of our proposals.

List of Publications

Journal Article

1. Rania Ben Halima, Slim Kallel, Walid Gaaloul, Zakaria Maamar and Mohamed Jmaiel *Toward a Correct and Optimal Time-aware Cloud Resource Allocation to Business Processes*, Future Generation Computer Systems, 2019 (to appear). (Impact factor: 5.768)
2. Rania Ben Halima, Slim Kallel, Mehdi Ahmed Nacer and Walid Gaaloul, *Optimal business process deployment cost in cloud resources*, The Journal of Supercomputing, 2020 (to appear). (Impact factor: 2.157)

Conference Proceeding

1. Rania Ben Halima, Imen Zouaghi, Slim Kallel, Walid Gaaloul and Mohamed Jmaiel, *Formal Verification of Temporal Constraints and Allocated Cloud Resources in Business Processes*, 32nd IEEE International Conference on Advanced Information Networking and Applications, AINA 2018, Krakow, Poland, May 16-18, 2018, 952–959 (Ranking: B).
2. Rania Ben Halima, Slim Kallel, Walid Gaaloul, and Mohamed Jmaiel, *Scheduling Business Process Activities for Time-Aware Cloud Resource Allocation*, On the Move to Meaningful Internet Systems, OTM 2018 Conferences- Confederated International Conferences: CoopIS, C&TC, and ODBASE 2018, Valletta, Malta, October 22-26, 2018, Proceedings, Part I, 445–462 (Ranking: A).
3. Rania Ben Halima, Slim Kallel, Walid Gaaloul, and Mohamed Jmaiel, *Optimal Cost for Time-Aware Cloud Resource Allocation in Business Process*, 2017 IEEE International Conference on Services Computing, SCC 2017, Honolulu, HI, USA, June 25-30, 2017, 314–321 (Ranking: A).
4. Rania Ben Halima, Slim Kallel, Kais Klai, Walid Gaaloul, and Mohamed Jmaiel, *Formal Verification of Time-Aware Cloud Resource Allocation in Business Process*, On the Move to Meaningful Internet Systems: OTM 2016 Conferences- Confederated International Conferences: CoopIS, C&TC, and ODBASE 2016, Rhodes, Greece, October 24-28, 2016, Proceedings, 400–417 (Ranking: A).

Table of contents

List of Tables	15
List of Figures	17
Introduction	21
1.1 Research Context	21
1.2 Objectives and Contributions	24
1.3 Road Map	26
2 Preliminaries	27
2.1 Introduction	27
2.2 Business process	28
2.2.1 Business process temporal constraints	28
2.2.2 Process Modeling	29
2.3 Cloud computing	31
2.3.1 Generalities	32
2.3.2 Cloud pricing strategies	33
2.3.3 CloudSim	35
2.4 Model Driving Engineering	38
2.5 Formal Verification	40
2.6 Mathematical Programming	44
2.6.1 Classification of optimization problems	45
2.6.2 Complexity of optimization problems	45
2.6.3 Optimization problem resolution	45
2.6.4 Linear programming	47
2.7 Conclusion	48
3 The Proposed Approach	49
3.1 Introduction	49
3.2 Motivation and Problems Statement	50
3.2.1 How to support cloud resource allocation in time-aware BP models?	50
3.2.2 How to ensure the correctness of time-aware BPs in cloud resources?	51
3.2.3 How to find the optimal deployment cost of time-aware BPs in cloud resources?	52
3.3 Case study	54
3.4 Proposed Approach	56
3.4.1 Graphical modeling	58

3.4.2	Verification of the temporal correctness of cloud resources allocation	58
3.4.3	Optimization of the BP deployment cost	59
3.5	Conclusion	60
4	State of the Art	61
4.1	Introduction	61
4.2	Temporal constraints and Cloud resources in BP	62
4.2.1	Formal specification in BP	62
4.2.1.1	Activity temporal constraints specification	62
4.2.1.2	Resource perspective specification	64
4.2.1.3	Time and resource specification	67
4.2.1.4	Synthesis	67
4.2.2	Formal verification	69
4.2.2.1	Formal verification methods	69
4.2.2.2	Structural verification	70
4.2.2.3	Temporal verification	71
4.2.2.4	Resource allocation verification	72
4.2.2.5	Time and resource allocation verification	75
4.2.2.6	Synthesis	77
4.3	Optimization of resource allocation	78
4.3.1	Resource allocation	80
4.3.1.1	Resource assignment	80
4.3.1.2	Resource scheduling	81
4.3.2	Cloud resource allocation	83
4.3.2.1	Cloud resource assignment	85
4.3.2.2	Cloud resources scheduling	87
4.3.3	Pricing strategies	90
4.3.4	Synthesis	91
4.4	Conclusion	93
5	Supporting Cloud Resources Temporal Constraints in BPs	95
5.1	Introduction	95
5.2	Graphical Modeling	96
5.2.1	Formal definitions	96
5.2.1.1	Cloud resources in BP	97
5.2.1.2	Business process model	98
5.2.2	BPMN extension	99
5.3	Model transformation	102
5.3.1	Transformation: from BPMN to timed automata	102
5.3.2	Automatic transformation	107
5.3.3	UPPAAL Meta-Model	109

5.4	Correctness analysis	109
5.5	Evaluation	111
5.5.1	Supporting pricing strategies description	111
5.5.2	BPMN model transformation	112
5.5.3	Checking CTL properties	115
5.6	Conclusion	118
6	Optimization of Business Process Deployment Cost in Cloud Resources	121
6.1	Introduction	121
6.2	Linear optimization	123
6.2.1	Inputs and decision variables	123
6.2.2	Problem constraints	124
6.2.3	BLP	127
6.2.4	MIP	127
6.3	CloudSim simulation	128
6.3.1	CloudSim extension	128
6.3.2	Unified Description Model	129
6.3.3	Simulation of resource allocation	130
6.4	Evaluation	131
6.4.1	Case study	131
6.4.2	Performance Analysis	132
6.4.2.1	Data inputs	133
6.4.2.2	Penalties Price Evaluation	133
6.4.2.3	AND Split/Join Constraints	135
6.4.2.4	Temporal Flexibility Constraint	135
6.4.2.5	Deadline Constraint	136
6.4.3	Comparison	136
6.4.4	Impact of the verification step	138
6.4.4.1	Impact of inputs	139
6.4.4.2	Impact of correct allocations' number	140
6.4.5	CloudSim Results	141
6.5	Conclusion	143
	Conclusion and Future Works	145
7.1	Fulfillment of objectives	145
7.2	Future Works	147
	Appendices	149
A	Implementation frameworks	151
A.1	Eclipse Modeling Framework: EMF	151
A.2	BPMN2 Modeler palette extension using Graphiti	152

B Plug-in creation	155
B.1 Meta-model creation	155
B.2 Java code generation	155
B.3 The dependencies of the extension plug-in	157
Bibliography	161

List of Tables

3.1	Process activities' temporal constraints and needs in cloud resources	55
3.2	Virtual machine instance properties by pr_{i1} =Amazon EC2	55
3.3	Virtual machine instance properties by pr_{i2} =Microsoft	55
4.1	Summary of the literature study of perspectives specification in BP	68
4.2	Summary of the literature study of verification in BPs	79
4.3	Summary of the literature study of resource allocation optimization	92
4.4	Summary of the literature study of human allocation optimization	93
4.5	Summary of the literature study of cloud resource allocation optimization	94
5.1	Temporal availabilities of cloud resources	98
6.1	Assignment result	132
6.2	CloudSim cost estimation	133
6.3	Data Input Ranges	133

List of Figures

1.1	Various pricing strategies to deploy BP model	23
1.2	Temporal verification and optimization in the BP lifecycle (adapted from [1])	25
2.1	Relative temporal constraints	30
2.2	Categories of BPMN object elements [2]	31
2.3	BP designed in BPMN	32
2.4	Conceptual reference model of cloud computing (defined by NIST) [3]	33
2.5	Amazon EC2 pricing strategies [4]	36
2.6	Basic architecture of Cloudsim [5]	37
2.7	Overview of ATL transformational approach [6]	40
2.8	Model checking process	41
2.9	Main elements of a timed automaton for UPPAAL	43
3.1	Verification-related research problem	52
3.2	Optimization related to research problem	53
3.3	Supervision service business process in BPMN	54
3.4	Approach overview	57
4.1	Time patterns proposed in [7]	63
4.2	Treatment process [7]	64
4.3	The Ralph language [8]	65
4.4	Process of patient examination [8]	66
4.5	Example process model with deadlock structural conflict [9]	70
4.6	BP model mapped into BPMN model [10]	72
4.7	Process model: Opening a bank account [11]	74
4.8	Static resource analysis approach overview [12]	75
4.9	Priority-based scheduling of process instances under human resource constraints [13]	83
4.10	Resource allocation of publish a book process [14]	84
4.11	Scheduling of a workflow with required resources [15]	88
5.1	The extension of resource element in BPMN	101
5.2	BPMN temporal extension	101
5.3	Allocation of R_1 as an on-demand instance cloud-resource to activity a_1	104
5.4	Resource allocation with R_1 as spot block to an activity a_1	104
5.5	Allocation of R_1 as a spot instance with an interruption risk to activity a_1	106
5.6	Allocation of R_1 as a spot block shared between activities a_1 and a_9 .	107
5.7	UPPAAL meta-model extension	110
5.8	Extended BPMN process with BPMN2 modeler	112
5.9	Pricing strategies specification	112

5.10	Pricing strategies specification	113
5.11	ATL transformation	114
5.12	Process activities timed automata	116
5.13	Resources timed automata	117
5.14	UPPAAL verifier's outcomes	118
6.1	Gantt chart of the service supervision process	132
6.2	Penalties prices	134
6.3	AND Split/Join Variation	135
6.4	Flexibility Evaluation	136
6.5	Deadline evaluation	137
6.6	Approaches Comparison	138
6.7	Impact of inputs on computation time	140
6.8	Impact of inputs on objective function	140
6.9	Impact of correct allocations' number	141
6.10	CloudSim Evaluation	142
6.11	Difference in % between our linear programs and the simulator	142
A.1	Generating Java code from an Ecore model	153
A.2	The BPMN2 Modeler extended palette	153
B.1	Ecore model and the DocumentRoot class	156
B.2	The loaded BPMN meta-model	157
B.3	Genmodel file and the generated packages	157
B.4	Dependencies required for the plug-in extension	158
B.5	The target runtime element	158
B.6	The model element	159

List of Acronyms

BP	Business Process
PAIS	Process Aware Information System
BPM	Business Process Management
BPMN	Business Process Modeling and Notation
ATL	ATLAS Transformation Language
BLP	Binary Linear Program
MIP	Mixed Integer Program
AInT	Activities Inflexible Temporal
AFT	Activities Flexible Temporal
IT	Information Technology
SaaS	Service as a Service
PaaS	Platform as a Service
IaaS	Infrastructure as a Service
AWS	Amazon Web Service
VM	Virtual Machine
FCFS	First Come First Served
MDE	Model Driven Engineering
CTL	Computation Tree Logic
GUI	Graphical User Interface
TA	Timed automata
TD	Temporal Dependency
UDM	Unified Description Model

Introduction

Contents

1.1	Research Context	21
1.2	Objectives and Contributions	24
1.3	Road Map	26

1.1 Research Context

Since the beginning of the nineties, Business Processes (BPs) have gained increased significance for almost any business. In order to manage and improve the quality and effectiveness of their operations, organizations have been more and more aligning their information systems in a process-centered way [16]. Along this trend, Process Aware Information Systems (PAISs) have emerged to better manage and execute operational processes involving people, applications, and/or information sources on the basis of process models [17]. Business Process Management (BPM) are examples of such systems [1, 18–20].

BP models are the main instrument of BPM. They represent BPs in terms of activities and their order. Various graphical notations have been proposed for BP modeling including Business Process Modeling and Notation (BPMN) [21], Event-driven Process Chain (EPC) [22], Yet Another Workflow Language (YAWL) [23], UML [24], etc. In fact, process modeling is part of the process design and analysis phase, which is the initial phase of a BP’s lifecycle in a PAIS [16]. After designing the BP model, it is important to use verification techniques to analyze it. By doing so, errors can be detected at an early stage and re-design efforts can be avoided. Then, once the BP is correctly (re)designed, it is automated into an operational/executable process to put it into practice. After deploying it on a PAIS, the BP is executed according to the BP model. In the end, process re-design can be performed to make improvements that have been identified after the analysis of process execution during the process diagnosis stage.

The BP field is influenced by a wide range of temporal constraints which rise from legal, regulatory, and managerial rules. A temporal constraint is a condition to control the system’s behavior over time. It specifies restrictions that occur across time [25, 26]. Consequently, time is a key resource for BPs within organizations and the satisfaction of temporal constraints such as deadlines is essential for a large set of BPs. Namely, offering products or services within restrictive deadlines helps organizations to raise their profits. For example, the aviation industry, the e-health, and the e-banking processes, are highly dependent on temporal constraints, since the violation of such constraints may lead to critical situations and could even threaten the safety of the involved parties [10]. Besides, activities require resource capacities expressed in terms

of memory amount (RAM) and CPU. So, organizations can deploy their BPs in cloud resources to satisfy activities requirements.

Motivated by the need of adopting flexible and cost effective BPs, organizations are looking for available services outside of them to quickly adapt to new business requirements and also reduce process development and maintenance costs. Cloud computing is recently gaining momentum due to its capability of outsourcing service-based BPs based on a scalable pay-per-use model. It is an attractive operational model allowing organizations to reduce upfront investment on Information and Communication Technologies and to tap into hardware and software resources of cloud providers in return of a fee. The cloud is known for resource elasticity and a pay-per-use model making it perfect for organizations that witness a surge of activities during particular periods of the year. For instance, at Christmas of 2017, Amazon.com had to temporarily cope with 280 millions¹ online retail transactions calling for immediate provisioning of resources that luckily were released once the load went back to normal.

In today's economic world, attracting and retaining customers constitutes a challenge. Indeed, many cloud providers offer competitive pricing strategies to accommodate users' changing and last-minute demands. However, this price variation puts more pressure on cloud providers who, for example, need to ensure resource availability on a short-notice. Organizations that wrestle with time, like shipping, need to respond quickly to any unforeseen event and hence, need to be able to call upon cloud providers anytime. Indeed, cloud computing allows organizations to optimize their BPs thanks to different techniques like virtualization and load balancing. However, this optimization should not happen on the expense of, for example, increasing operation costs [27] and/or violating time constraints. Striking the right balance between cloud resources' pricing strategies and BPs' temporal constraints is one of the BP designer challenge (Figure 1.1). The prices of cloud resources are variable and depends on temporal perspectives. Therefore, configuring a real cloud environment using a wrong estimation can lead to a waste of efforts, time, and money. As a result, to ensure that a BP is successfully executed, organizations would pay extra fees. Researchers often rely on simulation tools to model the mechanisms and evaluate their outputs [28]. That is why, the BP designer needs a cloud simulator in order to simulate a BP deployment in cloud resources and to get its real cost before deploying or even purchasing these resources from cloud providers. More precisely, he needs to: (i) graphically design a BP enriched with temporal constraints and cloud resources with their pricing strategies, (ii) ensure the temporal correctness of time-aware BPs, (iii) find the optimal-deployment cost of these BPs, and (iv) simulate the deployment of these BPs as well.

On the one hand, previous research in the field of BPM focused mainly on activities' temporal constraints [29–31] solely. Moreover, various research studies on the subject of cloud resource allocation in the BPM context [27, 32–34] consider only resource properties such as elasticity and shareability. In general, extensions of BP

¹www.hitwise.com.

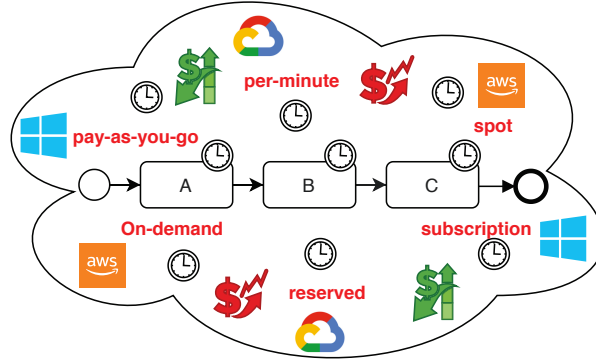


Figure 1.1: Various pricing strategies to deploy BP model

models with the representation and the definition of activities' temporal constraints and cloud resources were proposed in [27, 29, 30, 33, 35], separately. Nevertheless, less attention has been paid to cloud resources' temporal availability. In consequence, there is a clear lack of formal and explicit description and representation of pricing strategies that are specified in a natural language. Therefore, temporal correctness of cloud resource allocation can not be formally verified. Thus, during BP design and analysis phases, there is a need to consider not only the satisfaction of activities' temporal constraints but also the matching between the temporal constraints of activities and cloud resources.

On the other hand, several research studies focus on the resource allocation management in the BPM field [8, 27, 36–42], whereas, others deal with optimal *cloud* resource assignment or activities' scheduling in the cloud [14, 15, 43–58]. In addition, one key perspective when dealing with BPM is time [30]. However, to the best of our knowledge, optimizing the deployment cost of time-aware BPs based on pricing strategies has not been handled, yet. Therefore, before resource purchasing and BP deployment (i.e., during the implementation phase), there is a need to find the cloud resource assignment and/or the BP activities scheduling, that provides the optimal BP deployment cost, in the cloud while considering advanced temporal constraints and the diversity of pricing strategies. Moreover, in literature, different cloud simulators' extensions are proposed such as TeachCloud [59], CPEE [60], and CloudExp [61]. To the best of our knowledge, none of the existing cloud simulators can provide an optimal and real BP deployment costs in cloud resources while taking into consideration various pricing strategies. Considering the importance of cost optimization for time-aware BPs during the design and analysis phase, proposing an extension to this phase with BP cost optimization and simulation that also considers various pricing strategies along with temporal constraints on activities becomes a necessity.

1.2 Objectives and Contributions

In the light of the aforementioned shortcomings, the core objectives of this thesis are **improving the support of temporal constraints of both activities and cloud resources, as well as pricing strategies in BPs and providing solutions to minimize the BP deployment cost**. More precisely, our objectives are:

- to offer a modeling graphical tool to model/design graphically time-aware BPs deployed in cloud resources that are proposed under different pricing strategies;
- to ensure temporal correctness of cloud resource allocation in time-aware BPs;
- to find the resource assignment and/or the BP activities' scheduling that provides the optimal BP deployment cost;
- to simulate the best resource allocation to provide a more real BP deployment cost.

Our work aims to achieve our objectives by proposing three contributions. **Our first contribution**, comes in two steps: Establishing formal definitions [62] and proposing BPMN extension [63]. In the first step, we formally define time-aware BPs, cloud resources, and their pricing strategies. We consider various constraints related to BPs and cloud resources. In the second step, we extend BPMN to integrate cloud resource allocations in time-aware BP models. Thus, the BP designer is able to add cloud resources and their pricing strategies, as well as BP constraints. We developed a plug-in that takes into account our proposed extension.

A BP designer can still, however, make errors at design time which can lead to temporal violations at runtime. To deal with this problem, we propose our **the second contribution**, which comprises two steps: Model transformation [62, 64] and Correctness analysis [62, 64]. In the first step, we develop a set of rules to transform a time-aware BPMN model into a network of timed automata. In the second step, we formally verify this network against advanced properties known in the community as liveness, deadlock free, and deadline to check the temporal correctness of cloud resource allocation in BP. Finally, we implemented our rules to automate the transformation step using the ATLAS Transformation Language (ATL) as a model transformation language. We also use UPPAAL for the verification of some advanced properties [63].

For **our third contribution**, which also comes in two steps: Linear optimization and Simulation. Our aim is to find the optimal BP deployment cost in cloud resources and to simulate this allocation to have an estimation of its real cost. In the first step, we formulate our problem as two linear programming models defined through an objective function subject to a set of constraints. The objective function seeks to minimize the BP deployment cost. Constraints such as temporal constraints,

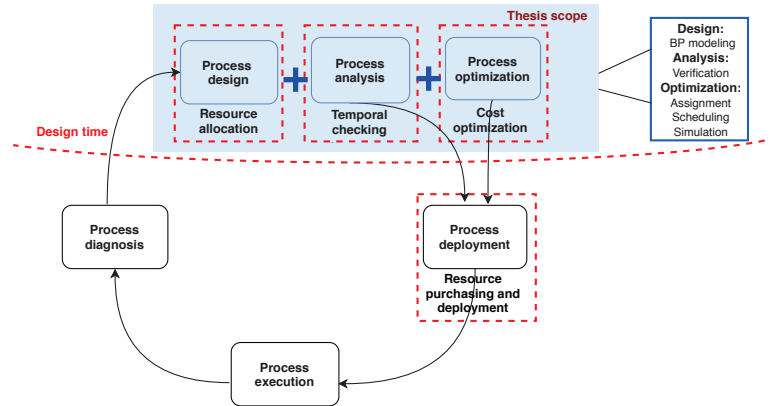


Figure 1.2: Temporal verification and optimization in the BP lifecycle (adapted from [1])

resources constraints, etc, limit the search space and help to converge to an optimal solution. Both linear programming models, (i.e., Binary Linear Program (BLP) and Mixed Integer Programming (MIP) models), provide an optimal BP deployment cost. They take as inputs a BP and a set of cloud resources proposed under various pricing strategies. If the activities' temporal constraints are flexible, using our MIP model, we extract the optimal BP scheduling plan [65,66]. Otherwise, using our BLP model, we provide the optimal assignment of cloud resources to deploy a BP [64,67]. Due to the high complexity of linear programs, i.e., NP-hard [68], handling BPs with large number of activities turned out cumbersome and inefficient. Therefore, to reduce their complexities, our linear models take as inputs a business process and a set of possible allocation options verified using our verification steps. Consequently, our approach is able to deal with business processes composed of large number of activities. In the second step, we extend the famous cloud simulator provided on the market, CloudSim [69], to support the simulation of the cloud resources consumed in the BP model and to compute its cost based on cloud providers APIs. Experiments have proven the effectiveness, feasibility, and scalability of our proposals.

Figure 1.2 shows the different phases of the cyclic lifecycle of a process model (design, deployment, execution, diagnosis). It illustrates the scope of this thesis which includes two stages. First, the *process design and analysis* phase is extended with resource and temporal perspectives in order to integrate the formal description of cloud resources and to prove the temporal correctness of cloud resource allocations. Second, this phase is enhanced with an optimization phase in order to provide the optimal BP deployment cost.

1.3 Road Map

This doctoral thesis is divided into 7 chapters:

- **Chapter 2: Preliminaries** gives the necessary background information by introducing BPs, activities temporal constraints, process modeling, cloud computing, model driving engineering, formal verification as well as mathematical programming.
- **Chapter 3: The Proposed Approach** starts by presenting the motivation and problem statement. Next, it presents our motivating example and then our proposed approach to optimize the BP deployment cost in cloud resources.
- **Chapter 4: State of the Art** positions our work by reviewing existing works in the literature related to our research fields.
- **Chapter 5: Supporting Cloud Resources Temporal Constraints in BPs** describes our first and second contributions. It presents in detail the 4 steps proposed to integrate the resource allocations time-aware BP models. We start by the modeling of resource allocation. Next, we present our extension to BPMN 2.0 notation, namely our BPMN 2 modeler plug-in for BP design. Then, we present a set of rules, as well as their implementation, which we define to automatically transform BPMN models into a network of timed automata models. In the fourth step, we rely on the model checking technique to verify the temporal correctness of our cloud resource allocation in BP.
- **Chapter 6: Optimization of BP Deployment Cost in Cloud Resources** presents our third contribution which comprises mathematical formulations and a CloudSim extension. In this chapter, we evaluate our solutions using experiments to prove their effectiveness, feasibility, and scalability.
- **Chapter 7: Conclusion and Future Works** concludes this thesis by summarizing the presented contributions and discussing potential future extensions.

Preliminaries

Contents

2.1	Introduction	27
2.2	Business process	28
2.2.1	Business process temporal constraints	28
2.2.2	Process Modeling	29
2.3	Cloud computing	31
2.3.1	Generalities	32
2.3.2	Cloud pricing strategies	33
2.3.3	CloudSim	35
2.4	Model Driving Engineering	38
2.5	Formal Verification	40
2.6	Mathematical Programming	44
2.6.1	Classification of optimization problems	45
2.6.2	Complexity of optimization problems	45
2.6.3	Optimization problem resolution	45
2.6.4	Linear programming	47
2.7	Conclusion	48

2.1 Introduction

This chapter presents the basic preliminaries and background needed for the understanding of our contributions described in the remainder of this manuscript. In Section 2.2 we present BP temporal constraints and modeling language. Next, Section 2.3 introduces cloud computing generalities, pricing strategies, and CloudSim which is the widely used software framework for modeling and simulation of cloud computing environments. Then, we present in Section 2.4 model driving engineering method, and in Section 2.5 formal verification technique that we used to transform BPMN models in order to be verified. Finally, Section 2.6 presents mathematical programming techniques, especially linear programming models to formulate our optimization problems.

2.2 Business process

This section presents, first, BP temporal constraints that we consider in our work (Section 2.2.1). Second, we introduce process modeling languages such as BPMN utilized to design BPs in our work (Section 2.2.2).

2.2.1 Business process temporal constraints

Both researchers and experts in business administration have long been interested in specifying the processes of organizations. The aim behind this interest is to understand, analyze, and thus improve such processes [70,71]. The BPs are influenced by a wide range of temporal constraints, which rise from legal, regulatory, and managerial rules [30]. Time perspective is a critical dimension to consider as it is closely related to customer satisfaction and cost reduction. In fact, delivering goods and/or services on-time raises the customer satisfaction level. Besides, an organization that efficiently manages time can reduce costs. The temporal constraints need to be viewed from multiple perspectives namely, the *relative* and *absolute* temporal constraints [72] that can be inflexible (i.e., tied to a specific time point) or flexible [29].

Hereafter, Based on [29–31, 73], we present activities temporal constraints that we consider in our work. Particularly, we classify temporal constraints into relative (Figure 2.1) and absolute temporal constraints.

- **Relative temporal constraints:** refer to activity duration and dependency:

- *Duration:* defines the completion time of an activity expressed as an interval $[MinD_{a_q}, MaxD_{a_q}]$. Let $s(a)$ (resp. $e(a)$) be the starting (resp. the ending) time of an activity a . Let $MinD_{a_q}$ and $MaxD_{a_q}$ be two relative time values representing respectively the minimum and maximum durations of a . Duration is defined as follow:

$$Duration(a, MinD_{a_q}, MaxD_{a_q}) \stackrel{\text{def}}{=} MinD_{a_q} \leq e(a) - s(a) \leq MaxD_{a_q}$$

For instance, depending on its severity and the patient's state, activity a (ovarian cancer surgeries) takes 1 to 10 hours. So, the interval [1h, 10h] presents the duration of the ovarian cancer surgeries where $MinD_a=1$ hour and $MaxD_a=10$ hours [7].

- *Temporal Dependency (TD):* is a relationship between two activities, a_q and a_l ($l \neq q$), in which one activity depends on the start or finish of another in order to begin or end [31]. We consider four temporal dependencies as follow;
 - * *Start-To-Finish (SF):* a_l can not finish until a_q starts within a time interval
 - * *Start-To-Start (SS):* a_l can not begin before a_q starts within a time interval

- * *Finish-to-Start (FS)*: a_l can not begin before a_q ends within a time interval
- * *Finish-To-Finish (FF)*: a_l can not finish until a_q has finished within a time interval

For illustration, Finish-to-Start dependency between two activities a_q and a_l , is defined as follows:

$$TD(\text{FS}, a_q, a_l, D_{min}, D_{max}) \stackrel{\text{def}}{=} D_{min} \leq s(a_l) - e(a_q) \leq D_{max}$$

This definition denotes that a_l should start its execution no later than D_{max} time units and no earlier than D_{min} time units after a_q ends. For instance, the contrast medium has to be accomplished at least 2 hours and at most 5 hours before the radiological examination takes place. Thus, we can specify a temporal dependency between activity a_1 (contrast medium completion) and activity a_2 (radiological examination) of type TD(FS, a_1 , a_2 , 2h, 3h) [7].

- **Absolute temporal constraints:** define a punctual temporal structure and refer to start and finish times of an activity. These temporal constraints can be inflexible (i.e. tied to specific time point) or flexible [29]. The inflexible temporal constraints (*AInT*) are Must Start On (MSO) and Must Finish On (MFO). They indicate the exact time, in which an activity must be scheduled to begin or complete. For example, the activity Ship products must start (respectively finish) at 8pm (respectively at 10pm) [10]. Thus, MSO(Ship products)=8pm (respectively MFO(Ship products)=10pm). Otherwise, a flexible temporal constraint does not specify a specific time point for a process or an activity, but rather imposes scheduling upper and/or lower bounds [29]. The flexible temporal constraints (*AFT*) considered are:

- *Start No Earlier Than (SNET), Finish No Earlier Than (FNET)*: indicate the earliest possible time that an activity can begin or complete. For example, the activity a (Receive shipment) has to start (respectively finish) no earlier than 9am (respectively 10am). So, SNET(a)=9am (respectively FNET(a)=10am).
- *Start No Later Than (SNLT), Finish No Later Than (FNLT)*: indicate the latest possible time when this activity is to begin or complete. For instance, the activity a (Receive shipment) has to start (respectively finish) no later than 6pm (respectively 9pm). Thus, SNLT(a)=6pm (respectively FNLT(a)=9pm).

2.2.2 Process Modeling

“BPs are what companies do whenever they deliver a service or a product to customers” [74].

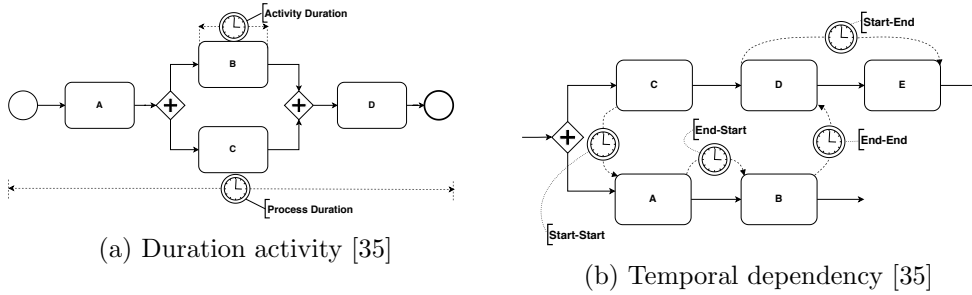


Figure 2.1: Relative temporal constraints

The BP modeling refers to creating BP models that describe the behavior of a BP according to its different perspectives. The first perspective is the control flow which describes the logical order between the process activities. The second one is the resource which describes the physical objects and human performers required to accomplish an activity. Data flow which describes the data exchanged between activities [75] represents the third perspective. Further, the fourth perspective of BP's behavior is time which is the key resource for processes within organisations [10]. In this thesis, we consider three perspectives: control flow, resource, and time.

Multiple graphical process modeling languages have been proposed such as BPMN [21], EPC [76], YAWL [23], UML [24] activity diagram, etc to design BPs. Despite their variances in expressiveness and modeling notations, they all share the common concepts of activities, events, gateways, artifacts and resources, as well as relations between them, such as transition flows [77]. The broadly accepted BP modeling language is Business Process Model and Notation (BPMN), therefore, we propose to take it as our stating modeling language [78].

BPMN was first released in 2004 by the Business Process Management Initiative (BPMI) [21]. It is a standard for BP modeling that permits to create and document process models. Known as defacto process modeling notation [79], it is widely used in the industry. Actually, BPMN has been enhanced with executable semantics enabling the execution of the modeled processes [80].

BPMN offers a rich set of elements that allows to capture various BP perspectives at different levels of detail. The BPMN elements can be categorized into a core set which contains the basic elements to model a BP and an extended set which contains more specialized elements to specify more complex business scenarios [81]. In fact, as shown in Figure 2.2, BPMN defines various constructs grouped into: *Flow objects*, *Connecting objects*, *Swimlanes*, and *Artifacts*. *Flow objects* are the main graphical elements to define the behavior of a BP. They include activities, events, and gateways. An activity is a generic term for work that is performed in a BP. It is a rounded rectangle with a name that describes the task to perform. The types of activities are: sub-process and task. An event is something that happens during the process execution. There exist 3 types of events: *Start*, *Intermediate*, and *End* events which

may be specialized to Message, Error, etc [80]. An event is graphically represented as a circle. A gateway is used to model the splits and joins in the process model. The various behaviors in a BP can be represented by three main types that are: *AND* (parallel synchronisation), *XOR* (exclusive choice and merging), and *OR* (inclusive choice and merging). Further, in BPMN there exist other specialized gateways such as event-based gateway, and complex gateways. The latter can be mapped to one of the three main types *OR*, *AND* or *XOR*. Connecting objects group includes *Sequence flow* element used to connect *flow objects*. In this manner, the order in which the activities will be executed in a process can be specified.

The *Artifacts*, *Swimlanes*, and other elements in *Connecting objects* allow to model the resource and data perspectives in the process [82]. *Swimlanes* group includes *Pools* and *Lanes* elements that permit to group a set of activities that are executed by a specific role. Besides, adding more informations to process model can be done through *Artifacts*. For instance, *Data object* element in *Artifacts* can be connected to activities through *Association* element in *Connecting objects*.

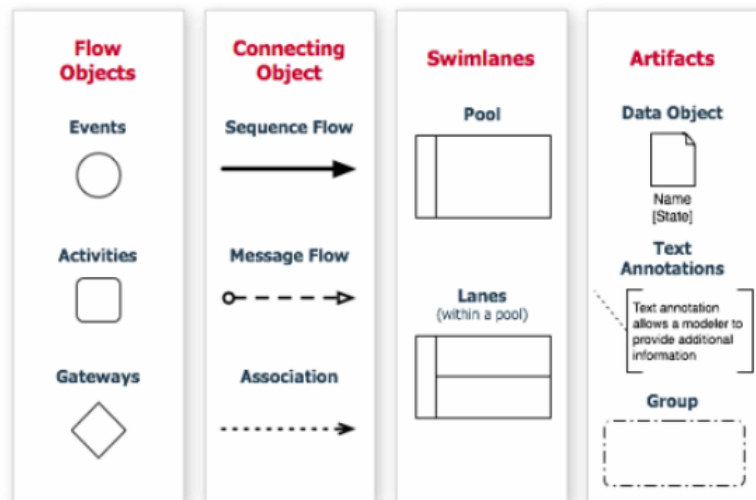


Figure 2.2: Categories of BPMN object elements [2]

We present in Figure 2.3 an example of a BP designed in BPMN and composed of: 3 activities: A, B, and C, and 2 parallel gateways.

2.3 Cloud computing

In this section, we give an overview of cloud computing environment and market.

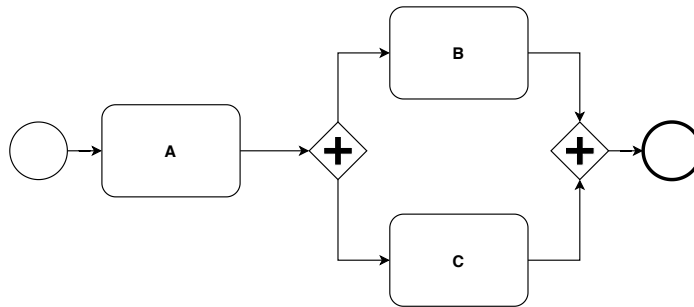


Figure 2.3: BP designed in BPMN

2.3.1 Generalities

Cloud computing has received a great deal of attention in a short period of time as an emerging paradigm in Information Technology (IT). It is defined by the National Institute of Standards and Technology (NIST) as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services). Those computing resources can be rapidly provisioned and released with minimal management effort [3]. In fact, it is an attractive operational model for organizations that wish, among many reasons, to avoid huge upfront investment on IT infrastructure and to outsource some (sometimes critical) operations to third parties. Cloud is known for resource elasticity and pay-per-use benefits making it perfect for organizations that witness intensive activities during particular periods of the year, i.e., before Christmas and Thanksgiving. Such a temporary peak would require rapid availability of resources that are released once the load returns to normal. Figure 2.4 presents an overview of cloud computing conceptual reference model defined by NIST. Various actors (cloud consumer, cloud provider, cloud carrier, cloud auditor, and cloud broker), activities, and functions in the cloud are included.

Cloud delivers different types of resources between applications at three various layers offered as services: at the highest layer *Service as a Service (SaaS)* which refers to providing the services of applications over the web on as needed basis [83]; the middle layer *Platform-as-a-Service (PaaS)* provides an operational platform allowing customers to manage, develop and execute their applications; and at the bottom layer *Infrastructure-as-a-Service (IaaS)* where consumers may access to highly automated and scalable resources delivered as a service via the Internet [16]. IaaS offers *network*, *storage*, and *compute* resources. The next section presents cloud providers pricing strategies under which computing resources are proposed.

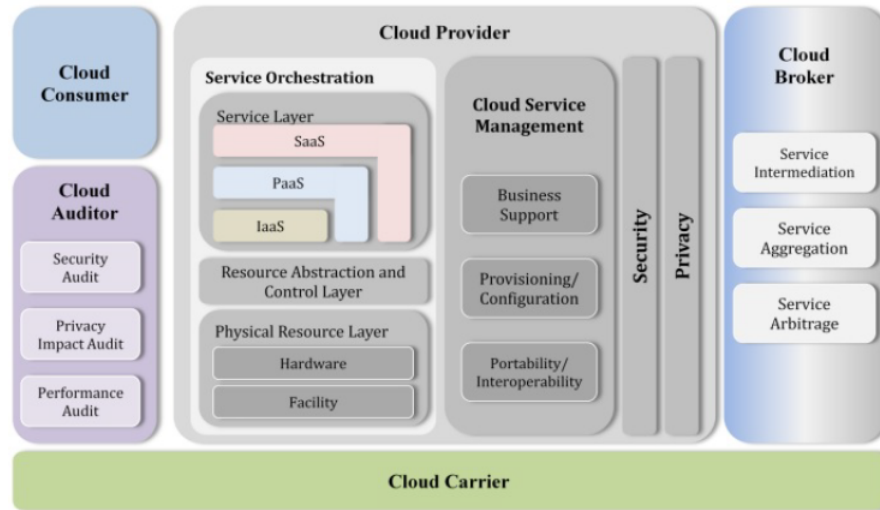


Figure 2.4: Conceptual reference model of cloud computing (defined by NIST) [3]

2.3.2 Cloud pricing strategies

Various types of computing resources with different QoS are offered by cloud providers, specifically IaaS providers, in various pricing strategies. This variety of QoS and pricing strategies also gives a high level of flexibility in resource management and consumption. Cloud resources' value is determined by cloud providers and is captured through pricing. Basically, this value embraces the economics and computer science research area. In fact, some studies handle the pricing issue using economics and business methods, while some others consider computer science techniques to tackle economic problems related to pricing. In our work, we focus on cloud resource pricing as an economic issue for computer science area.

Cloud providers take into consideration some factors to define the cost of the provided cloud resource. Indeed, the first factor is *the cost of the resource* in the data center in which an extra percentage is added to achieve the targeted profit. The cost of the resource includes the price of the physical machine, maintenance service cost, and electricity fees. The second factor is *the market competition*. To remain in business, cloud providers must be aware of prices for the same services by other providers in the marketplace and set their prices competitively. Cloud computing market is moving rapidly towards a highly competitive price [84]. Cloud providers need to measure the value and satisfaction given to their customers when consuming their cloud resources. For that, one factor to consider when determining a cloud resource cost is the *value to the customers*.

When cloud providers understand the cost of the cloud resource provided, the cost proposed by competitors for the same resource, and the value perceived for the

customers, it is time to figure out what type of pricing strategy to define [85]. The most commonly used pricing strategies in cloud markets, especially in IaaS cloud marketplaces, are on-demand (pay-as-you-go) and subscription-based pricing strategies.

There exist several cloud providers that sell cloud resources, mainly Google Cloud Platform (GCP) [86], Microsoft Azure [87], and Amazon Web Service (AWS) [4]. GCP tends to have an advantage in storage and network performance. Besides, its computing cloud service "Google compute engine" has no service's termination fees. Moreover, GCP does not rely on long-term contracts to keep customers. No upfront costs are required as well. Users do not need to make commitments to get discounts. However, GCP has the smallest amount of data centers. In addition, computing resources provided by GCP have better performance and flexibility compared to AWS, but less resources types variety.

Azure offers good performance and optimal network. But it gives a less specific selection of resources. Added to that, the primary approach to get discounts on Azure is that customers must have Microsoft enterprise agreement. Also, if clients want to work on a platform other than Windows Server operating system (a Microsoft product), Azure might not be the best solution. AWS is a secure cloud services platform. It offers products with many options for supporting the existing platforms (Linux, Windows, etc). Particularly, Amazon Elastic Compute cloud (EC2) is a web service that provides secure and resizable VMs in cloud [4]. It allows paying only for capacity that consumers actually use. EC2 has the advantage of auto scaling feature. It has the facility to change automatically the number of VMs that are running at peak time. But, AWS is considered complex as it provides various VMs' types with several costs. While cloud computing features vary among providers, the challenge is to supply users with good compute services for reasonable prices. Therefore, each cloud provider has its own pricing strategy. Indeed, customers pay in function of the time, quantity, or VM's instance type they consume. Pricing in the cloud context is similar to how we pay for utilities like water or electricity since users only pay for the services they consume. There are multiple pricing strategies for VM instances. We find that GCP, for example, provides two main strategies. It supplies with on-demand resources per-minute with a minimum of 10 minutes. It offers also a strategy called sustained use discounts. This means the longer the period of running an instance by the clients during each month, the larger the discount they will receive. Microsoft Azure also provides two main strategies. It charges customers per-minute if they use the on-demand (pay-as-you-go) strategy. Also, it proposes pre-paid subscription pricing strategy and gives a 5% discount for a minimum of 6000\$ commitment.

Whereas, AWS can charge customers per-hour. AWS instances can be purchased using three main pricing strategies which are on-demand, reserved, and spot instances. The latter are detailed in the following. We note that we focus on AWS, because it continues to lead the market in terms of maturity and offering the widest range of functionality, VMs, and pricing strategies.

- **On-demand:** the customer pays an hourly fixed amount with no long-term

commitment. The procured resource capacity can be increased or decreased depending on the applications' requirements and the payment is done for the procured capacity, only [4].

Figure 2.5a shows Amazon EC2 instances and their on-demand strategy cost. For instance, the on-demand strategy cost of m5.12xlarge is 2.664\$.

- **Reserved:** the customer can make a one-time, upfront payment for a long-term reservation of a resource capacity and pays a significantly hourly rate for running capacity instances in the future.

Figure 2.5b shows Amazon EC2 instances and their reserved strategy cost. For instance, the reserved strategy cost of m5.12xlarge is 1.865\$.

- **Spot instance:** customers bid for unused resource capacities to secure some spots offered at a spot price [4] and with an interruption risk due to the bidding process. In response to potential disruptions, Amazon also proposes spot instances with a specified duration (also known as **Spot blocks**) that are continuously available (from 1 to 6 hours) [62]. Figure 2.5c illustrates an example of a spot block pricing strategy. The hourly price of m5.12xlarge as a spot block instance available up to 6 hours is 1.497\$. Whereas, Figure 2.5d depicts a spot instance pricing history presenting the variation of spot price during 1 day. For example, the spot price of m5.12xlarge from 6pm until 8pm is 0.67\$.

2.3.3 CloudSim

Cloud Computing systems are complex and need specific tools to analyze some quality concerns including resource provisioning, task scheduling, network configuration, security or virtual machines management. Testing in real world environment is one technique for evaluating the performances of a system, but it is a costly and time consuming method [88]. Moreover, the user is not able to access all the systems' components that need to be analyzed. But, with a simulation tool, he can focus on each quality, separately while considering many different scenarios. Cloud simulators provide a stable, cost-efficient and scalable environment, where tests can be replicated, repeated and validated by the research community. They empower users to control all the layers of the cloud system, videlicet the physical resources configuration, the infrastructure topology, the code middle-ware platform, the cloud application services and the user workload behavior [89].

CloudSim, the *most popular* and *sophisticated* cloud simulator available today, has been developed in 2010 by the CLOUDS Laboratory at the Computer Science and Software Engineering Department of Melbourne University (Australia). It is used in research by several universities and organizations [90]. The simulator enables modeling of CPU components, RAM, storage, Virtual Machine (VM), host, cloud broker and data center, as well as VM allocation policy, VM scheduler, dynamic

	vCPU	Memory (GiB)	Linux/UNIX Usage
m5.large	2	8	\$0.111 per Hour
m5.xlarge	4	16	\$0.222 per Hour
m5.2xlarge	8	32	\$0.444 per Hour
m5.4xlarge	16	64	\$0.888 per Hour
m5.12xlarge	48	192	\$2.664 per Hour
m5.24xlarge	96	384	\$5.328 per Hour

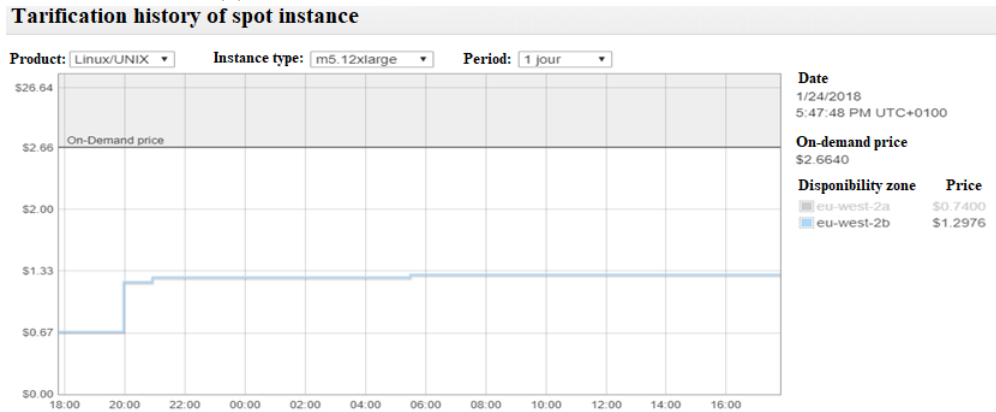
(a) On-demand strategy

STANDARD 1-YEAR TERM					
Payment Option	Upfront	Monthly*	Effective Hourly**	Savings over On-Demand	On-Demand Hourly
No Upfront	\$0	\$1361.45	\$1.865	30%	\$2.664
Partial Upfront	\$7779	\$648.24	\$1.776	33%	
All Upfront	\$15246	\$0	\$1.740	35%	

(b) Reserved strategy

	1 hour	6 hours
m5.12xlarge	\$1.152 per Hour	\$1.497 per Hour

(c) Spot blocks strategy



(d) Spot instance strategy

Figure 2.5: Amazon EC2 pricing strategies [4]

workload, etc. CloudSim is open source and has been developed in java programming language. Moreover, it allows a user to model and simulate all the cloud infrastructure resources. It also requires much less effort and time to implement applications in cloud environment.

The CloudSim framework consists of three layers as shown in Figure 2.6 (bottom-up description):

- **The Core Simulation Engine:** Queuing and processing of events, management of cloud system entities such as host, VMs, brokers, etc.
- **CloudSim:** Representation of network topology, delay of messages, VM provisioning, CPU, storage and memory allocation, etc.
- **User code:** Cloud scenarios, user requirements, user Broker, application and workload configurations, allocation and scheduling policies declarations.

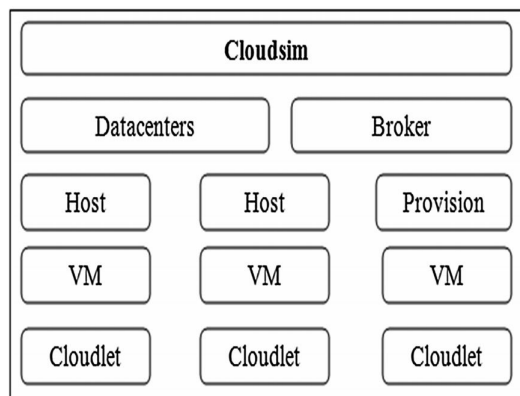


Figure 2.6: Basic architecture of Cloudsim [5]

As shown in Figure 2.6, the main components of CloudSim [89] are: *Datacenters*, *Hosts*, *Virtual Machines (VM)*, *Broker*, and *Cloudlets*. *Datacenter* class models the core infrastructure level services (hardware, software) offered by resource providers in the cloud computing environment. The *Datacenters* encapsulate a set of *Hosts* and their resource configurations (memory, cores, capacity, and storage). Furthermore, every *Datacenter* component instantiates a generalized resource provisioning component that implements a set of policies for allocating bandwidth, memory, and storage devices. Each *Host* component can instantiate multiple *VMs* and allocate cores based on predefined processor sharing policies. Each *VM* has an owner, which can submit *Cloudlets* to the *VM* to be executed. The *Broker* class is responsible for mediating negotiations between SaaS and cloud providers and such negotiations are driven by QoS requirements. It acts on behalf of applications. This class must be extended

for evaluating and testing custom brokering policies. The *Cloudlet* class models (i.e., defines specific attributes such as length of instruction, input/output filesize, no of processor required, etc) the cloud-based application services (program based tasks) such as content delivery, social networking, etc. CloudSim implements the complexity of an application in terms of its computational requirements [91].

CloudSim implements different scheduling policies [92] such as:

- **SpaceShared:** schedules one task on virtual machine at a given instance of a time and after its completion it schedules another task on virtual machine. This same policy is used to schedule the virtual machines on the host. This policy behave same as the first come first serve algorithm (FCFS) [93].
- **TimeShared:** schedules all tasks on virtual machine at the same time. It shared the time among all tasks and schedule simultaneously on the virtual machine. This policy is also used to schedule the virtual machine on the host. The concept of round-robin (RR) scheduling algorithm [93] is used in this policy

2.4 Model Driving Engineering

This section gives a brief overview of the Model Driven Engineering (MDE) method used to bridge from timed BP models, expressed in BPMN 2.0, to a formal language model, expressed in timed automata.

Currently, software systems are evolving. This factor makes development and maintenance of systems more complex than before. Also, several issues have appeared and enhanced throughout the years such as the increasing complexity of software, concerns separations, business separations, multiplicity of needs, and platforms. Therefore, software no longer concentrates only on the code. As a solution, modeling in computer science and engineering appeared to master the complexity of software. A model is a simplified representation of an aspect of the world for a specific purpose.

MDE [94–96] has been used in the development process. MDE is a software development methodology that focuses on creating and exploiting domain models. It aims to raise the level of abstraction in program specification to increase automation in program development. The idea promoted by MDE is to reuse models at different levels of abstraction for developing systems. Therefore, by the abstraction of technologies' implementation we evolve more easily toward new technologies. Besides, it increases productivity by maximizing compatibility between systems, simplifying the design process, and boosting communication between teams working on the system. Other goal of MDE is the separation of concerns. There are two main concerns: (i) business concerns that are the core of the application and (ii) implementation platforms. Also, there are several other possible concerns like security, user interface, quality of service, etc. We argue that each concern is designed by a model. The integration of these concerns can be whether by transformation, fusion or weaving of models. To have a better productivity, models have to be well defined by the concept

of meta-model. As a result, they can be manipulated and interpreted using tools with different meta-models that can be processed simultaneously. Meta-model is a modeling methodology used in software engineering. It typically defines the language and processes from which to form a model. Briefly, it is a model of a model. For example, it can define concretely a modeling language.

An increase of automation in program development is reached by using executable model transformations. Model transformations are at the heart of MDE, and provide the essential mechanism for manipulating and transforming models automatically. A model-to-model transformation is done via transformation rules which describe the mapping between source model and target model. There are two types of transformations. First, the endogenous transformation, which is in the same technological space i.e., the source and target models are conform to the same meta-model. Second, the exogenous transformation, which takes place between two different technological spaces i.e. the source and target models are conform to different meta-models. There exist several model transformation languages. The most popular ones are ATL [97,98], QVT [99] and Epsilon [100]. We depict ATLAS Transformation Language (ATL) since it is the most mature language between the others. ATL is a hybrid model transformation language containing a mixture of declarative and imperative constructs based on Object Constraint Language (OCL) [101] for writing expressions. However, the declarative programming style is more recommended. It is sometimes difficult to provide a complete declarative solution for a given transformational problem. In that case, developers may resort to the imperative features of the language [98]. Declarative programs are context-independent. They only declare what the ultimate goal is, but not the intermediary steps to reach that goal. Furthermore, the same program can be used in different contexts, which is difficult to do with imperative programs that often depend on the context. An ATL module consists of four main elements which are header, import, helpers and rules:

- **Import:** this section is optional and defines the ATL libraries to be imported.
- **Header:** introduces the name of the module developed and the meta-models involved. The target meta-model declaration is introduced by the create keyword. The source meta-model is introduced by the keyword from.
- **Helpers:** are variables or functions used to implement code that can be reused to avoid code redundancy.
- **Transformation rules:** describe the transformation and contain OCL expressions. The rules are composed of two parts. The “from” part indicates the elements of the source model to be transformed. The “to” part contains transformation expressions that indicates the elements of the target model. There are three different kind of rules which are matched rules, called rules and lazy rules.

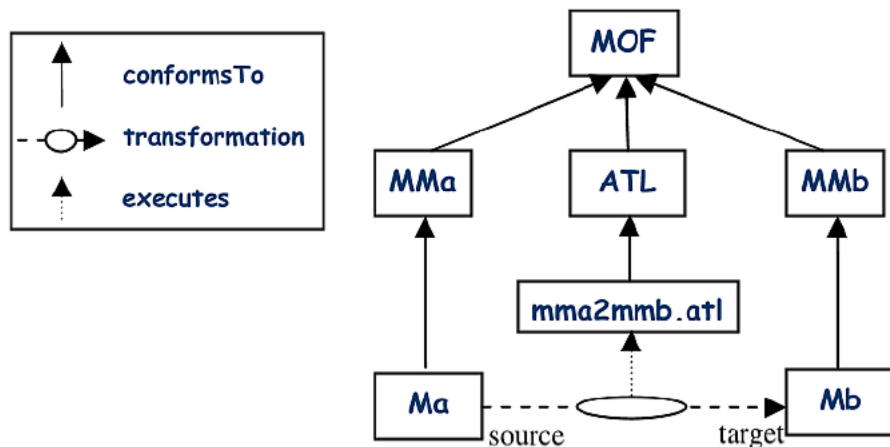


Figure 2.7: Overview of ATL transformational approach [6]

As shown in Figure 2.7, ATL is applied in a transformational pattern [98]. It provides ways to produce a set of target models from a set of source models. More specifically, in this pattern, a source model Ma is transformed into a target model Mb . The transformation is driven by a transformation definition `mma2mmb.atl` written in the ATL language [98]. The transformation definition is a model. The source and target models as well as the transformation definition are conform to their metamodels respectively MMa , MMb , and ATL . The meta-models are conform to the Meta-Object Facility (MOF) meta-meta-model [97].

2.5 Formal Verification

Hardware and software are widely used. They are where errors are unacceptable. Therefore, model verification is necessary to ensure the design correctness at an early stage. The use of formal verification methods has been considered, for a long time, very desirable for ensuring the correctness of a system's design. However, the complexity of these methods made them only accessible to specialists. Thus, they were only used for very critical systems. With the appearance of model checkers, formal verification becomes accessible for "normal" engineers. In fact, formal verification can be realized with a model checker to verify whether the modeled system satisfies some given properties. We note the existence of several model checkers such as UPPAAL [102,103], TINA [104], ProB [105], etc. The process of model checking is depicted in Figure 2.8. It consists of three main steps which are modeling, specification of properties, and formal verification.

The first step (i.e., (1) in the Figure 2.8) is modeling the system. In order to formally verify a model, it must first be converted into a simpler verifiable format called

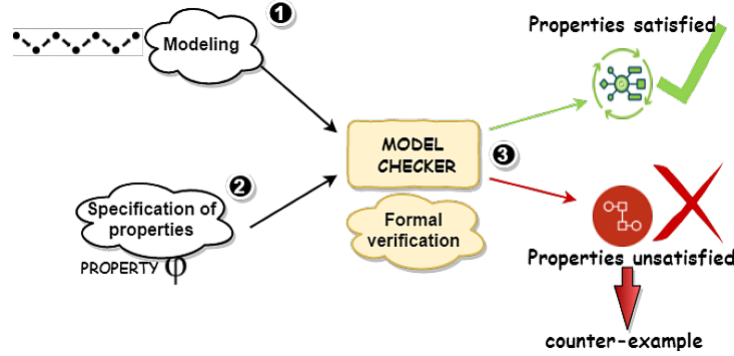


Figure 2.8: Model checking process

formal language such as timed automata [106], Petri nets [107], process algebra [108], Event-B [109], etc.

In our work, we use timed automata language while it offers a formal syntax and semantics. Furthermore, timed automata are nowadays a widely used modelling formalism with rich theoretical foundations and a number of developed verification tools like UPPAAL [110], KRONOS [111], etc. A number of case studies demonstrate that timed automata are a suitable formalism for modelling systems of industrial sizes and the tools have already reached a reasonable degree of maturity and efficiency. A combination of an easily understandable syntax and semantics together with the support for Clike constructs and data structures (e.g., in the tool UPPAAL) makes this a widely applicable and successful approach for the modelling and verification of time dependant systems.

A timed automata is a directed graph where nodes correspond to states (also called locations) of the system and edges correspond to transitions between these nodes [112]. In [110], a timed automata is extended with a finite set of real-valued variables, named clocks, for modeling time passing. A clock is represented as a non-negative real number. To define conditions over clocks, constraints are used [113]. They permit to reset values of certain clocks to zero. X is a set of non-negative clocks. A clock constraint, called invariant, is associated with each state. It has to be satisfied so, that, the system remains in the current state [112]. The transitions are labelled by both temporal constraints, called guards, and clock variables, and are synchronized through binary channels. Guards and invariants are conjunctions of constraints $x \bowtie v$, where x is a clock in X , $v \in \mathbb{R}^+$, and $\bowtie \in \{<, \leq, =, >, \geq\}$. A transition is enabled if the guard is evaluated to true and the source state is active. $\Psi(X)$ is the set of constraints over X [62].

Definition 2.5.1. A *timed automata* is a tuple (L, X, l_0, Tr, I) where:

- L is a finite set of states,

- X is a finite set of clocks,
- l_0 is an initial state,
- $Tr \subseteq L \times \Psi(X) \times 2^X \times L$ is the set of transitions,
- $I: L \rightarrow \Psi(X)$ is a function that assigns invariants to states.

Definition 2.5.2. A transition is a tuple $tr=(l, \alpha, \psi, cl, l') \in Tr$ where l is a source state, l' is a target state, α is a label, ψ is a guard, and cl is a set of clocks to reset.

Consider the timed automaton of Figure 2.9 with one clock t which gets set to 0 each time the system moves from *Start* to *A1* location. The invariant ($t \leq 2$) associated with the location *A1* ensures that switch from *A1* to *End* happens within time 2. Resetting another independent clock y together with the b -labeled switch from $s1$ to $s2$ and checking its value on the d -labeled switch from $s3$ to $s0$ ensures that the delay between b and the following d is always greater than 2. Notice that in the above example, to constrain the delay between a and c and between b and d the system does not put any explicit bounds on the time difference between a and the following b , or c and the following d . This is an important advantage of having multiple clocks which can be set independently of one another.

Figure 2.9 illustrates the graphical representation of timed automata designed with UPPAAL. The graph nodes are locations: *Start*, *A1*, and *End*. The edges are transitions between locations. For instance, the edge between *Start* and *A1* contains a clock t (initialized to zero) and a synchronization variable of type sender (*start!*). Clock reset is called update with UPPAAL. The timed automata contains also an invariant ($t \leq 2$) to specify the requirement that the system can not stay in location *A1* for at most 2 units, and a switch must occur before the invariant is violated. Whereas, the edge between *A1* and *End* contains a guard ($t \leq 0 \ \&\& \ t \geq 2$) to ensure that the transition between both locations is enabled only if the clock t is at least 0 unit and at most 2 units. Further, the variable channel *done*, of type receiver, is used for synchronization.

The second step (i.e., (2) in Figure 2.8) is properties' specification that we give to the model checker using properties' languages in order to decide whether some property holds for a timed automaton. The expression of properties in UPPAAL is by Computation Tree Language (CTL) [114,115], which is a specification language for finite state systems. The specific types of properties that can be expressed with CTL can be classified as:

- **Liveness:** a specific condition is guaranteed to hold eventually, i.e., "something good will happen (eventually)" [116].

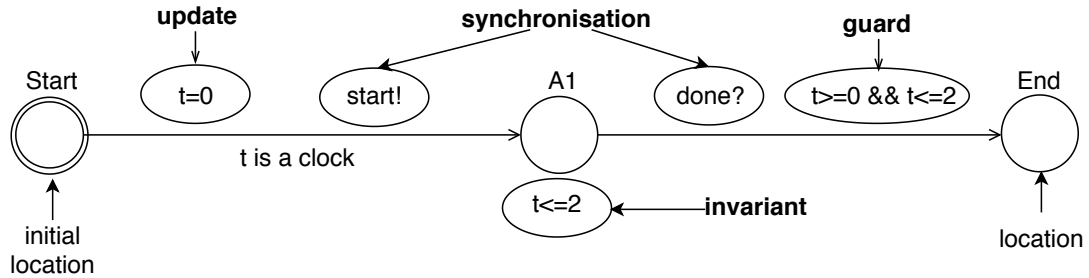


Figure 2.9: Main elements of a timed automaton for UPPAAL

- **Deadlock freeness:** "the system never ends up in a state where it cannot perform any action" [117]. So, we verify that the system is deadlock free expressed using a special state formula proposed by the model checker UPPAAL.
- **Deadline:** it express the set of states where the corresponding action is expected to be executed without delay.

UPPAAL is composed of a Graphical User Interface (GUI) for modeling systems and a model checker engine to verify if the system is correct with respect to a set of properties. UPPAAL GUI has three types:

- **The system editor:** allows the user to describe and edit timed automata system which consists of global declarations, timed automaton templates, process assignment, and a system definition section.
- **The simulator:** allows the user to virtually interact with the system described. The simulator shows the system state by displaying the states of automata and the values of variables. It simulates the system to validate that it behaves as intended and to see how certain states are reachable [110]
- **The verifier:** accepts the user's formulated properties to be verified on a particular timed automata model, and displays the result of verification: true or false depending on whether the property was satisfied or not.

Third step (i.e., (3) in Figure 2.8) is the verification of the given properties using the query language for UPPAAL which is a simplified version of CTL. The properties' checking is searching in automata the locations, paths, and circuits that have the given characteristics. It generates the space of all possible states and exhaustively checks the properties that hold the possible dynamic behavior of the model. Each CTL operator is a pair of symbols. The first member of the pair is either A (along All paths), or E (there Exists a path). The second member of the pair is one of the following X (neXt state), G (Globally in the future), F (in a Future state), or U (Until). The CTL logical operator are \vee , \wedge , \neg , and \Rightarrow mean respectively *and*, *or*, *not*,

and *imply*. Indeed, the CTL formulas, which can be analyzed with UPPAAL, are the following:

- **AG** ψ : along All paths the property ψ holds Globally
- **EG** ψ : there Exists a path where the property ψ holds Globally
- **AF** ψ : along All paths the property ψ holds at some state in the Future
- **EF** ψ : there Exists a path where the property ψ holds at some state in the Future

In UPPAAL, these CTL formulas are implanted as follows:

- **A []** ψ : is the implementation in UPPAAL of the formula AG ψ
- **A <>** ψ : is the implementation in UPPAAL of the formula AF ψ
- **E []** ψ : is the implementation in UPPAAL of the formula EG ψ
- **E <>** ψ : is the implementation in UPPAAL of the formula EF ψ

UPPAAL model checker offers multiple ways to help in the verification of the given properties. Examples of such ways are counter-example generation and visualization. In fact, verification step makes sure that the properties are satisfied (the process is safe, deadlock-free, deadline meet, and locations are reachable). In case that properties are unsatisfied, a counter-example is given showing under which circumstance the error can be generated.

2.6 Mathematical Programming

The term “programming” in the context of mathematical programming means planning activities that consume resources and/or meet requirements. Mathematical programming is mostly known as a technique used by decision makers to mathematically formulate optimization problems and develop optimal values of the decision variables [118].

Planning and scheduling problems are formulated as mathematical model based on equalities and inequalities. The solution satisfying all these constraints is considered as acceptable plan or schedule. Mathematical programming enables us to formally define optimization problem through variables, objective functions and constraints [119].

2.6.1 Classification of optimization problems

The classification of optimization problems is defined based on several criteria. The nature of the objective function and the constraints are an important criteria for this classification. Optimization problems can be classified into four classes: linear, nonlinear, quadratic, and geometric programming [120]:

- **Linear Programming (LP):** In this class, the objective function and the constraints are linear functions of variables. There exist different class of LP problems that are grouped based on the decision variables types. We can find Integer Linear Programming (ILP), where all variables are integer, Binary Linear Programming (BLP), where all variables are binary, and Mixed Integer Linear Programming (MILP) problems, where some variables are integer and others are binary.
- **Nonlinear Programming (NLP):** In this class, the objective function and/or the constraints are/is nonlinear.
- **Quadratic Programming (QP):** In this problem type, the objective function is quadratic and the constraints are linear.
- **Geometric Programming (GP):** In this type of problems, the objective function and the constraints are expressed as polynomials [121].

Optimization problems can be classified also based on the deterministic nature of the variables as deterministic and stochastic programming problems [120].

- **Deterministic Programming:** In this category of problem, all the decision variables are deterministic. If a model provides always the same output for a given input, it is considered deterministic.
- **Stochastic Programming:** When some or all the problem parameters are stochastic (random or probabilistic) variables.

2.6.2 Complexity of optimization problems

There are mainly two classes of optimization problems: P and NP [68]. P is the class of problems solvable by algorithms operating within a polynomial time (if it is run $O(n^k)$ steps where k is a constant and n denotes the problem size). NP is the class of problems solvable by non-deterministic algorithms operating within a polynomial time. It stands for a non-deterministic polynomial.

2.6.3 Optimization problem resolution

Two different methods can be applied to solve optimization problems, namely: exact methods and approximate methods.

- **Exact methods:** are usually used by researchers in case of small instances. Exact methods find the optimal solution and assess its optimality. There exist numerous exact methods such as the family of Branch-and-Cut, Branch-and-Bound, and Cutting planes. Exact methods are known to be time expensive, so they can not be applied to large NP-hard problems or difficult ones. Among these methods, we can cite, mainly, the linear programming methods, the gradient descent, etc.

A linear programming method identifies a linear program that models the multi-criteria problem where the objective function and the constraints are all linear. Then, this linear program can be solved efficiently by certain algorithms such as:

- *Branch-and-Bound:* is based on the principle of enumerating the solution space of a given problem and then choosing the best solution [122]. The enumeration has a tree structure. Each node of the tree separates the search space into two sub-spaces, until the complete exploration of the solution space [123]. The Branch-and-Bound algorithm is named also simplex algorithm.
 - *Cutting-Plane:* its basic idea is to cut off parts of the feasible region of the linear program relaxation, so that the optimal solution becomes an extreme point and therefore can be found by the simplex method. It is the first algorithm developed for integer programming that could be proved to converge in a finite number of steps. Even though the algorithm is considered not efficient, it has provided insights into integer programming that have led to other, more efficient, algorithms [124].
 - *Branch-and-Cut:* is a method of great interest for solving various combinatorial optimization problems. This method is a result of the integration between two methods: (i) cutting plane method, and (ii) branch-and-bound method [125]. The cutting planes lead to a great reduction in the size of the search tree of a pure branch and bound approach. Therefore, a pure branch and bound approach can be accelerated by the employment of a cutting plane scheme [126].
- **Approximate methods:** are often used when instances become too large for exact methods. They are based on random and iterative approaches using heuristics and/or meta-heuristics algorithms to solve optimization problems. An approximation algorithm is guaranteed to run quickly (in time polynomial in the input size) and to produce a solution for which the value of the objective function is quantifiably close to the optimal value.
Heuristics are search methods produced based on human's intuitive and creative thinking, and are often useful in local search to find good solutions quickly in a restricted area. Metaheuristics are higher level heuristics that control the whole process of search, so that global optimal solutions can be obtained systematically and efficiently. Although metaheuristics cannot always guarantee to

obtain the true global optimal solution, they can provide very good results many practical problems. They combine exploration and exploitation while looking for acceptable solutions. Usually, metaheuristics can enhance the computing power of a computer system greatly without increasing the hardware cost. Various heuristic and metaheuristic algorithms were proposed in the literature such as Simulated Annealing [127], Genetic Algorithm [127], Ant Colony [127], etc.

2.6.4 Linear programming

In our work, we formulate our optimization problems as linear programming models known for its simplicity, easy way of understanding, and flexibility to analyze the problems. Thus, in the following, we focus more on linear programming model.

Linear programming does not have as much history compared with to other fields of mathematics, largely due to the difficulty in solving most problems without the aid of computer calculation. One of the first attempts to provide a feasible solution to linear programming problems was made by Joseph Fourier, who published a method in 1827. In the present, linear programming is more useful than ever before for management and logistical challenges due to easy access of large amounts of data. It should be noted that linear programming optimization problems are NP-hard. So they require high computational efforts to find out an optimal and even a feasible solution for large size problems. For that, combining with improved algorithms and improvements in computing power, linear programming is able to solve larger and more complicated problems. The primary challenge is accurately modeling applicable situations and using available data efficiently [128].

The first goal of linear program models is to minimize or maximize a linear function, called z , which is subject a finite set of linear constraints. The function z is known as the objective function and is a linear combination of the variables (e_1, e_2, \dots, e_n) with the general form $z=v_1e_1+v_2e_2+\dots+v_n e_n$ where each v is a constant. The linear constraints can be either equality constraints or inequality constraints and take the general form where a and b are constants [128].

$$a_1e_1+a_2e_2+ \dots + a_n e_n \begin{pmatrix} \geq \\ = \\ \leq \end{pmatrix} b$$

A feasible solution is any combination of the variables (e_1, e_2, \dots, e_n) that satisfies the constraints, and the set of these n -tuples is known as the feasible region. A problem which has no solution which satisfies all of the constraints is infeasible [128]. The full form of the general linear programming problem is as follows

$$\text{Minimize } v_1e_1+v_2e_2+ \dots + v_n e_n = z$$

$$\begin{aligned}
a_{11}e_1 + a_{12}e_2 + \dots + a_{1n}e_n &\geq b_1 \\
a_{21}e_1 + a_{22}e_2 + \dots + a_{2n}e_n &\geq b_2 \\
&\vdots \\
a_{m1}e_1 + a_{m2}e_2 + \dots + a_{mn}e_n &\geq b_m \\
e_i &\geq 0, i \in 0, 1, \dots, n \\
b_i &\geq 0, i \in 0, 1, \dots, m
\end{aligned}$$

Linear program is composed of decision variables that present the quantities to find. So, they are the unknowns variables of a mathematical programming model. Generally, they are presented using an algebraic form like e_1, e_2, \dots, e_n . n presents the number of decision variables and e_i presents the name of the i th variable. The n -tuple (e_1, \dots, e_n) satisfying the constraints of a linear program is a feasible solution of this problem. A solution that maximizes or minimizes the objective function of the problem is called an optimal solution.

As mentioned in Section 2.6.3, a linear program can be solved using methods including: Branch-and-Bound, Cutting-Plane, or Branch-and-Cut to converge to an optimal (exact) solution. However, since it belongs to the NP-hard optimization problems, it requires high computational efforts to find out an optimal and even a feasible solution for large size problems. For that, it is often more important to reduce the search space to avoid waiting for a long time to obtain an approximate (near optimal) solution.

2.7 Conclusion

This chapter provided the background that helps position our contributions. In fact, it introduced BP's temporal constraints that are considered in our work and Business Process Modeling Notation (BPMN) as an example of BP modeling language. Then, we presented cloud computing generalities, pricing strategies, and the most popular cloud simulator "CloudSim". Next, we described the MDE methodology that helps to transform informal BP modeling language BPMN into a formal BP modeling, i.e., timed automata which has been introduced in Section 2.5. Finally, we presented optimization problems that we use in our work as mathematical model to find optimal cost.

The Proposed Approach

Contents

3.1	Introduction	49
3.2	Motivation and Problems Statement	50
3.2.1	How to support cloud resource allocation in time-aware BP models?	50
3.2.2	How to ensure the correctness of time-aware BPs in cloud resources?	51
3.2.3	How to find the optimal deployment cost of time-aware BPs in cloud resources?	52
3.3	Case study	54
3.4	Proposed Approach	56
3.4.1	Graphical modeling	58
3.4.2	Verification of the temporal correctness of cloud resources allocation	58
3.4.3	Optimization of the BP deployment cost	59
3.5	Conclusion	60

3.1 Introduction

Time and resource management in organizations aims to reduce cost and maximize profits. As mentioned in Chapter 2, most of the pricing strategies are defined based on temporal constraints. Thus, a cloud resource has a limited temporal availability that restricts the time span allowed for its use at a defined cost. As a result, BP designers and researchers are seeking, not only to have support of activities' temporal constraints, cloud resources, and pricing strategies in current BPM suites, but also to ensure the temporal correctness of cloud resource allocation in time-aware BP models. Besides, the variety of cloud resources, pricing strategies, and activities temporal constraints does not help the BP designer to easily find the optimal BP deployment cost. For this reason, the BP designer faces many challenges including: (i) the selection of the suitable cloud resource for each activity with the best pricing strategy, and in case of temporal flexibility (ii) the definition of the start and end times of BP activities as to match the temporal availability of the less expensive cloud resource. In other words, it is important to discover the best allocation that minimizes the

BP deployment cost while respecting a set of constraints such as time. Consequently, specifying and managing' activities temporal constraints, cloud resources, and pricing strategies in the BP field is becoming a hot research topic [27, 30–33, 48, 129].

Despite this interest, no one, to the best of our knowledge, has studied how to support cloud resources' temporal constraints in the BPM context. Besides, many efforts have been made to reduce the BP deployment cost in cloud computing, but currently none of the existing studies consider the variety of pricing strategies and advanced temporal constraints to deal with such an issue. On the basis of this challenge, we proposed an end-to-end approach, composed of three steps: *Modeling*, *Verification*, and *Optimization*, that aims to minimize the deployment cost of time-aware BPs in the context of cloud computing while considering different pricing strategies. The inputs of our approach are (i) a BP with its set of activities' requirements including time, RAM, CPU and (ii) a set of cloud resources delivered from a set of cloud providers under different pricing strategies. The output of our approach is a correct cloud resource allocation that optimizes the deployment cost of time-aware BPs.

This chapter is organized as follows. We present our motivation and problem statement in Section 3.2. Then, we present our case study in Section. 3.3. Next, an overview of the proposed approach is presented in Section 3.4. Finally, Section 3.5 concludes the chapter.

3.2 Motivation and Problems Statement

Our research work is motivated by the following three main issues: How to support cloud resource allocation in time-aware BP models?, How to ensure the correctness of time-aware BPs in cloud resources?, and How to find the optimal deployment cost of time-aware BPs in the cloud?

3.2.1 How to support cloud resource allocation in time-aware BP models?

A BP model is a procedural, semi-formal specification of the order in which activities should be executed to achieve a goal [130, 131]. Some BPs might be subject to some temporal constraints which can be relative or absolute. Such constraints require a set of capacities expressed in terms of RAM and CPU. For that, they can consume cloud resources that are proposed under various pricing strategies defined based on temporal constraints (e.g. Amazon pricing strategies such as reserved and spot).

As mentioned above, researchers do not focus enough on neither the modeling nor the graphical design of cloud resources' temporal constraints. In general, various studies [10, 29–31, 35, 132] have addressed time in a BPM context by integrating activities' temporal constraints description, representation, etc. Besides, in those studies, the authors extended the popular BP modeling standard BPMN to allow the BP designer to express temporal constraints such as deadlines and activity durations [133]. How-

ever, a formal definition and representation of cloud resources' temporal constraints is still missing. Although BPMN offers capabilities to specify resource perspective, it remains very poor and does not allow to specify temporal constraints of cloud resources related to pricing strategies.

In this thesis, we aim to extend the cloud resource perspective in BPs by considering cloud resources' temporal availabilities. Hence, we propose formal definitions of cloud resources and their pricing strategies in BPs taking into account temporal constraints. Next, we propose an extension of the BPMN 2.0 meta-model to support the pricing strategies as well as activities' temporal constraints. In order to achieve our goals, we need to answer the following research questions:

RQ1: How to model the cloud resource allocation in time-aware BPs?

RQ2: How to integrate cloud resources' temporal availabilities, their pricing strategies, and activities temporal constraints in BP models at design time?

3.2.2 How to ensure the correctness of time-aware BPs in cloud resources?

The deployment of a BP on a cloud infrastructure offers to the BP designer a pool of cloud resources delivered under competitive pricing strategies defined based on a temporal perspective. Therefore, the BP designer needs to verify the satisfaction of the BPs' time constraints in a cloud context. For instance, cloud resources' temporal availabilities may not cover activities' temporal durations. On one hand, some research studies [27, 32, 34] have addressed the verification of cloud resource allocation in BPs taking into account cloud resource properties such as elasticity and shareability. On the other hand, other studies [30, 31, 73, 134] have addressed the issue of verification of temporal constraints in BP models. Nevertheless, the formal verification of BPs' temporal constraints with respect to cloud resources' availabilities has been overlooked.

Verifying the correctness of cloud resource allocation in time-aware BPs is becoming a challenge. As shown in Figure 3.1, the BP designer models his time-aware BP model with the required cloud resources. The selected cloud resources can have limited temporal availabilities related to the selected pricing strategies. Thereby, he may make errors that might lead to temporal violation and damage the temporal correctness of the BP execution, especially in case of complex BP models.

The use of formal methods to model BPs and cloud resources can be very useful to validate and analyze the temporal correctness of cloud resource allocation. Basically, temporal violations can be detected before the deployment of cloud resources based on mathematical foundations offered by formal methods.

In this thesis, our objective is to ensure the correctness of a cloud resource allocation to time-aware BPs in order to avoid deadlock at runtime. Towards this end, we developed a set of rules to automatically transform time-aware BPMN models into a network of timed automata. Then, we formally verified this network against

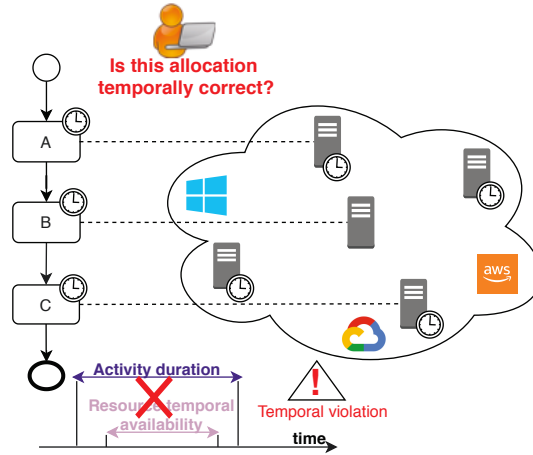


Figure 3.1: Verification-related research problem

advanced properties known in the community as liveness, deadlock free, and deadline. To achieve this objective, we list the following research questions:

RQ3: How to transform a BP model into a formal language?

RQ4: How to formally verify the cloud resource allocation in time-aware BPs?

3.2.3 How to find the optimal deployment cost of time-aware BPs in cloud resources?

The use of cloud resources still presents issues related to optimization due to the variety of pricing strategies and activities requirements. For instance, the BP designer can select cheaper cloud resources for activities to guarantee a minimal cost. However, this selection may not satisfy a set of constraints such as time. Besides, he has different allocation possibilities and does not know which is the best allocation in terms of cost and constraints satisfaction. Moreover, when the BP has some temporal flexibility, he can advance or delay the start and end times of activities in order to match the temporal availability of a cloud resource. But, he may not respect all the constraints. Figure 3.2 shows the above mentioned problems. In fact, a BP designer chooses a cloud resource allocation possibility in order to minimize the BP deployment cost. The selected resource allocation can have a cheaper cost but it violates a set of constraints. Therefore, before deployment, the BP designer needs to know if his allocation choice provides the optimal BP deployment cost and satisfies a set of constraints or not. In other words, he needs to find for each activity the suitable cloud resource with the best pricing strategy while satisfying constraints such as time, amount of memory, number of CPUs, etc. Until now, as far as we know, optimizing the deployment cost of time-aware BPs based on pricing strategies has not been well studied. Mainly, there exist some studies [132, 135, 136] on minimizing (non-cloud) resource allocation

cost under resource temporal constraints. Nevertheless, as cloud resources are always available, researchers do not consider temporal availabilities constraints, related to pricing strategies, to minimize the BP deployment cost in cloud resources.

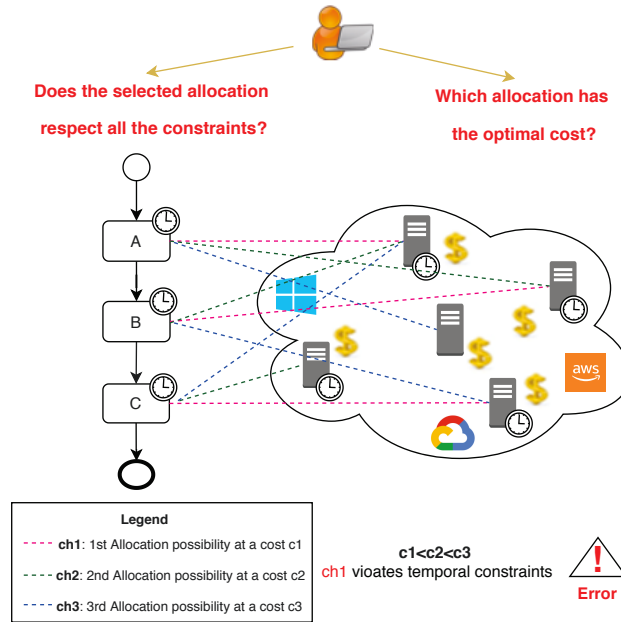


Figure 3.2: Optimization related to research problem

Resorting to mathematical programming techniques to formulate optimization problems can be very efficient to find an optimal cost. Mainly, an optimal and correct allocation can be discovered before deploying or purchasing cloud resources using a mathematical model defined via an objective function subject to a set of constraints.

In this doctoral research, our purpose is to discuss how the cost of BP deployment cost can be optimized. To this end, we need to formulate our optimization problem as a mathematical model by defining an objective function with constraints that would guide the optimization. To achieve our objective, we state the following research questions:

RQ5: How to find an optimal allocation of cloud resources to activities in a BPM context?

RQ6: How to determine in which order activities should arrive and leave resources to minimize the deployment cost?

3.3 Case study

We present, in the following, the BP taken from the French Telecommunication operator Orange [27] to illustrate and motivate our approach. It is also used to explain our approach in the next chapters.

First, we introduce a “service supervision” BP which presents how a customer’s complaint is addressed in case of a drop in the quality of service. Figure 3.3 is a model of the supervision process in *BPMN 2.0*. This BP is triggered when a complaint is sent by a customer using *Get service trouble ticket* activity “ a_1 ”, after which either necessary data are retrieved “ a_2 ” or the management test is started “ a_3 ”. Data retrieval can be performed automatically “ a_4 ” or via a script “ a_5 ”. As for the service test, it requires both “ a_6 ” and “ a_7 ”. Finally, by replying to the customer “ a_8 ” and troubleshooting the process “ a_9 ”, the process meets its end.

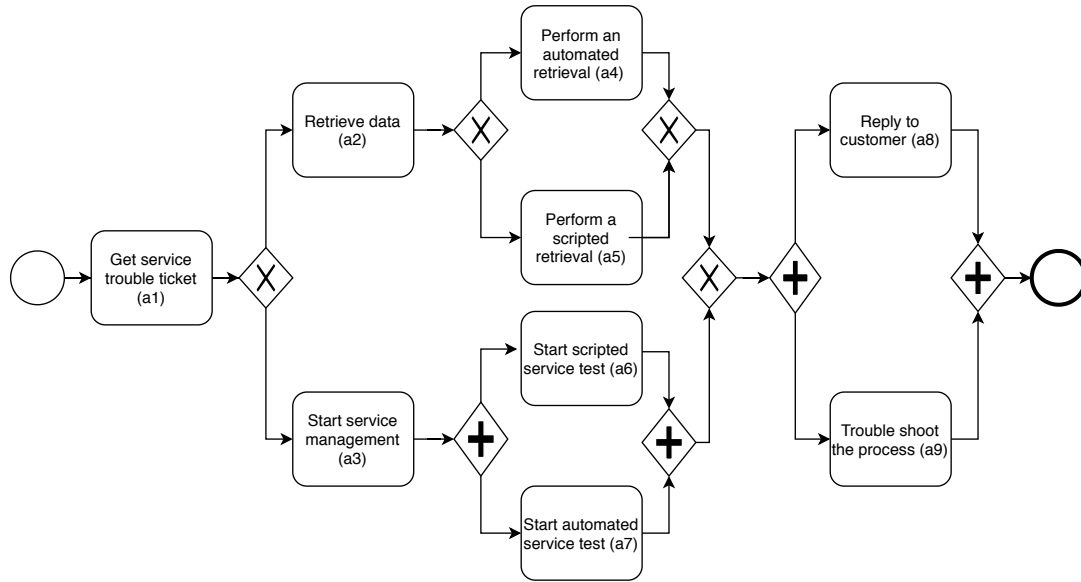


Figure 3.3: Supervision service business process in BPMN

Table 3.1 lists temporal constraints for the “supervision service” BP along with the capacities required by each activity in this BP, expressed in terms of *RAM* and *CPU*. For instance, the minimum duration for “ a_2 ” and “ a_6 ” is 2 hours and the maximum duration is 3 hours. Moreover, the time lag between the end of “ a_3 ” end and the start of “ a_7 ” should be between 2 and 5 hours. Also, activities such as “ a_2 ” require resources with 15GB of RAM and 2 CPUs. Besides, some activities are subject to cancellation penalty prices p_q [137, 138] that should be added to the BP’s deployment cost. For instance, “ a_1 ” has a financial penalty cost of 0.7\$ in case of its execution cancellation due to a resource interruption.

Table 3.1: Process activities' temporal constraints and needs in cloud resources

Activities									
	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9
Durations	[1h,2h]	[2h,3h]	[1h,1h]	[1h,4h]	[1h,2h]	[2h,3h]	[1h,2h]	[1h,1h]	[1h,2h]
Penalties in \$	0.7	0	0	0.2	0	0	0	0	0
RAM in GB	16	15	16	28	16	28	30	15	15
CPU number	4	2	2	8	4	8	8	2	2

Table 3.2: Virtual machine instance properties by pr_{i1} =Amazon EC2

Instances	RAM (GB)	vCPU	On-demand	Reserved (no upfront)	Spot block	Spot
R_1 = m4.xlarge	16	4	0.215\$/h	0.147\$/h	0.129\$/h [0h,1h]	0.0491\$/h [6pm,1am ⁽⁺¹⁾]
					0.142\$/h [1h,6h]	0.0386\$/h [1am,6pm]
R_2 = r3.large	15	2	0.166\$/h	0.105\$/h	0.096\$/h [0h,1h]	0.0225\$/h [3am,10pm]
					0.102\$/h [1h,6h]	0.0381\$/h [10pm,3am ⁽⁺¹⁾]
R_3 = m3.2xlarge	30	8	0.532\$/h	0.380\$/h	0.293\$/h [0h,1h]	0.0787\$/h [10am,9pm]
					0.372\$/h [1h,6h]	0.0863\$/h [9pm,10am ⁽⁺¹⁾]

Table 3.3: Virtual machine instance properties by pr_{i2} =Microsoft

Instances	RAM (GB)	vCPU Number	On-demand (c_{i21})
R_4 =A2m v2	16	2	0.128\$/h
R_5 =D4 v2	28	8	0.387\$/h

On one hand, the BP designer has different resource allocation options. For instance, while R_1 satisfies the resource requirements of “ a_1 ” and “ a_9 ” in terms of RAM and CPU, it can be assigned to both of activities as a spot block available for 1 hour up to 6 hours at a cost of 0.142\$/h. However, “ a_9 ” would not be correctly performed. Indeed, the temporal availability of R_1 does not cover the duration of the time lag between the start time of a_1 and the end time of a_9 (i.e., $2h+3h+4h+2h=11h > 6h$). Besides, since R_2 satisfies the capacity requirements of ‘ a_2 ’, the BP designer can allocate it as a spot instance that is available from 3 a.m to 10 p.m at a cost of 0.0225\$/h to perform a_2 . If the BP starts at 6 p.m, so, “ a_2 ” may start at 8 p.m and finish at 11 p.m. As a result, the selected pricing strategy for R_2 , assigned to “ a_2 ”, is not appropriate, i.e., $11 \text{ p.m} \notin [3 \text{ a.m}, 10 \text{ p.m}]$. Unfortunately, the choices of the BP designer can lead to a temporal violation. For this reason, he needs to avoid the mismatch between temporal constraints of both activities and cloud resources.

On the other hand, the BP designer has as objective to allocate the suitable cloud resource for each activity at a minimal cost. For that, he can assign, for example, R_1 and R_4 for “ a_2 ”, “ a_3 ”, “ a_8 ” and “ a_9 ” since they satisfy both their requested amount of RAM and CPU. Besides, a cloud resource has more than one price depending on its temporal availability [63]. Thus, an activity can consume one cloud resource under various pricing strategies that can be temporally available. In addition, the activities’ temporal constraints can be flexible. Therefore, the BP designer can specify the start and end times of BP activities to overlap with temporal availabilities of cheaper cloud resources. For instance, “ a_2 ” can start at 8 am and finish at 11 a.m so it consumes R_1 as a spot instance available from 1 am until 6 p.m. It can, however, also start at 8 p.m and finish at 11 p.m. Thus, it consumes R_1 as a spot block. Consequently, there are various options of resource assignment and scheduling. We can conclude that it is hard to find (i) the suitable cloud resource to allocate for each activity, (ii) the best pricing strategy for each cloud resource, especially for activities subject to cancellation penalties prices, and (iii) the optimal scheduling plan of a BP in case of flexible temporal constraints.

3.4 Proposed Approach

In this section, we present an overview of our end-to-end approach (Figure 3.4) for a correct cloud resource allocation that provides the optimal BP deployment cost. It takes as inputs a BP, activities temporal constraints, cloud resources, and pricing strategies. Three main contributions define our proposed approach: *Graphical Modeling*, *Verification* and *Optimization*. In the modeling step, the BP designer is able to design his BP enriched with cloud resources and temporal aspects. This is ensured by extending BPMN meta-model with cloud resources, pricing strategies, and temporal constraints. So, the BP designer will have a BP modeled in BPMN and enriched with activities temporal constraints, cloud resources, and their pricing strategies.

On the one hand, the BP designer may have erroneous allocations which lead

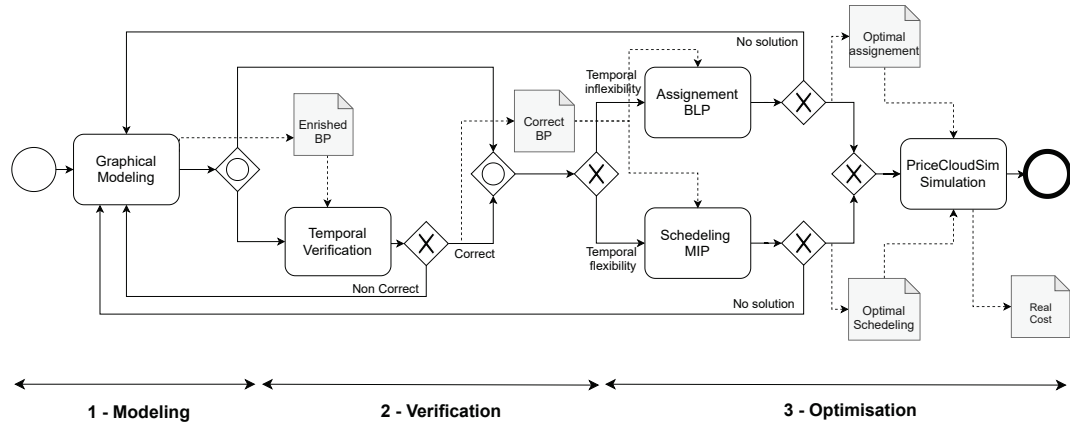


Figure 3.4: Approach overview

to runtime errors if left untreated at design-time. Therefore, we propose a formal verification step to avoid such issue. While BPMN is a semi-formal language, we propose to transform the extended BP models, designed in BPMN, into a formal language to formally verify the temporal correctness of a cloud resource allocation. Thus, if the temporal constraints are satisfied, the BP designer will have a correct BP modeled in BPMN. Otherwise, he should try another resource allocation and verify its temporal correctness.

On the other hand, the BP designer can not easily find the optimal BP deployment cost, because of the variety of activities requirements, cloud resources, and pricing strategies options. Namely, he has different allocation possibilities and each one provides a different BP deployment cost. To this end, we formalize the optimization problem as linear programming models to minimize the deployment cost of time-aware BPs. More precisely, we offer two solutions. The first one is an assignment approach and the second is a scheduling approach used when activities temporal constraints are flexible. Linear programming models are known with the need of long time to obtain optimal solution for problem with large size problems. As a result, we extend our optimization approach to deal with more complex and “large” BPs.

The price of cloud resources is variable and depends on temporal perspective such as spot instance. So, configuring a real cloud environment using a wrong estimation can lead to waste of efforts, time, and money. Thereby, to ensure that the BP is successfully executed organizations would pay extra fees. To overcome this challenge, researchers often rely on simulation tools to model the mechanisms and evaluate their outputs [28]. That is why, the BP designer needs a cloud simulator in order to simulate a BP deployment in cloud resources and to get its real cost. For that, we rely on an existing cloud simulator to simulate the optimal BP deployment in cloud resources and to provide its real cost.

3.4.1 Graphical modeling

In this section, we present our first contribution defined based on two steps: a modeling step and a BPMN extension step. As noted before, the resource perspective lacks a formal and unified description. Compared to other perspectives such as the control-flow and the organizational perspective, it reminds poorly operated. Some research efforts are made for the enhancement of the resource description in BPM [8, 139, 140]. There exists also previous works on the formal specification of activities temporal constraints [10, 30, 31]. However, enhancing the cloud resource description is still poorly managed due to the lack of a formal definition of pricing strategies, that are specified in natural language. To deal with such issue, we define a formal model of a BP enriched with temporal constraints, cloud resources, as well as pricing strategies. Namely, we define first, a formal model of BP that is subject to some temporal constraints that could be either relative or absolute. Then, we define formally a cloud resource and its pricing strategy specified based on temporal constraints. Next, we move to the BPMN extension step. BPMN is the broadly accepted BP modeling language and it offers a better understanding and shareability of BP models. Further, BP designers know very well this modeling language. So, we propose to take it as our stating modeling language, then to extend it to support our proposed model of cloud resources and pricing strategies. The aim of this BPMN extension is to help the BP designer to model his BPMN process enriched with cloud resources and temporal constraints in order to allocate correctly and cost-effectively the required cloud resources. We implement our approach as an Eclipse plugin which is an extension of BPMN 2 modeler [141] to offer to the BP designer a graphical modeling tool supporting the temporal constraints of cloud resources and activities as well as pricing strategies¹.

3.4.2 Verification of the temporal correctness of cloud resources allocation

In this section, we give an overview of our second contribution: the verification which is composed of two steps: Model transformation and Correctness analysis. To avoid the design errors such as temporal violation, there is a need to use formal methods to verify the correctness of cloud resource allocation in time-aware BP. Formal methods have proved their usefulness in the design of correct systems (e.g. timed automata [106], Petri nets [107], process algebra [108], Event-B [109], etc.). They promote the use of mathematical foundations and formal logic to specify and reason about system properties. So, adopting formal methods to formally specify a cloud resource allocation in time-aware BPs can be very effective to validate and check temporal constraints before deploying or even purchasing resources from cloud providers. While BPMN is a semi-formal language we transform the extended BP designed in BPMN into timed automata as a formal language.

¹<http://www-inf.it-sudparis.eu/SIMBAD/tools/BPMN4CP>

As reported in Section 2.1, timed automata is a suitable formalism for modelling and verifying time dependant systems. Besides, using timed automata allows us to analyze properties of our model using UPPAAL [102]. Thus, we define a set of transformation rules to generate from the extended BPMN model into a (network) timed automata model. These models are respectively conform to BPMN and UPPAAL meta-models. Nevertheless, the BP designer can have a limited knowledge about formal methods and languages, especially timed automata, so he can often make errors at the moment of manual transformation from BPMN models into timed automata models. Many research approaches have been proposed for generating formal models exploiting model-driven principles and techniques. Besides the advantage of conceptual simplicity, MDE leads to clear architecture, efficient implementation, high scalability and good flexibility. Thus, we resort to MDE-based model-to-model transformation language to conduct an automatic transformation to help the BP designer to provide a network of timed automata. More precisely, we implement our transformation rules using ATL, the widely adopted language for model transformation [142]. As a result, a BP, designed in BPMN, deployed in cloud resources purchased based on various pricing strategies, can be transformed automatically into a network of timed automata models.

Turning now to the correctness analysis step. Towards this end, the network of timed automata atomically generated is formally verified against advanced properties such as liveness, deadlock-freeness, and deadline using the model checking tool UPPAAL. The objective is to verify that there is a matching between the temporal constraints of both activities and cloud resources, so that allocating cloud resources to time-aware BPs is correct.

3.4.3 Optimization of the BP deployment cost

We present in this section our third contribution detailed in Chapter 6. Basically, we define our optimization problem using mathematical formulation. Indeed, we offer two solutions to optimize the deployment cost of the BP designed in the modeling step. Based on activities' temporal constraints flexibility, we propose a Mixed Integer Programming (MIP) model that provides an optimal execution plan for BP activities that can be optimally executed using cloud resources. Otherwise, if activities temporal constraints are not flexible, we propose a Binary Linear Program (BLP) that selects for each activity the suitable cloud resource, cloud provider, and pricing strategy. BLP and MIP models are defined through an objective function that minimizes the BP deployment cost under a set of constraints. In this work, we provide exact solutions and violating one of these constraints is not tolerated. Thus, if one constraint is not satisfied, so BLP and/or MIP do not have any solution. Therefore, the BP designer goes back to the modeling step. In summary, we help him to reduce the cost BP deployment while respecting a set of constraints such as activities requirements and resources constraints.

It is worth noting that linear programming optimization problems are NP-hard [143]. Hence, they require high computational efforts to find out an optimal and even a feasible solution for large size problems. For that, it is often more important to reduce the search space to avoid waiting for a long time to obtain the optimal solution. To deal with more complex and large BPs, before moving to the optimization step, we check in our verification approach the temporal correctness of cloud resource allocation in time-aware BPs. Then, we take as inputs for our linear programs: (i) a BP and (ii) only a set of cloud resources and pricing strategies that ensure correct resource allocations. In this manner, the size of the optimization problem is reduced and so the response time of our linear programs is reduced. Consequently, our linear programs converge in a short time to the optimal solution [144].

Cloud resources price is variable and timed constrained (e.g., spot instance). Thus, to avoid wasting time, effort, and money to purchase and configure a real cloud environment, we complete our approach with a simulation step to simulate the optimal BP deployment and to provide an estimation of its real cost. More precisely, we extend CloudSim [66], *EPriceCloudSim* [145] that supports AWS pricing strategies, to simulate a time-aware BP deployed in cloud resources. *EPriceCloudSim* takes as inputs a unified description model specified as an XML document composed of BP activities, temporal constraints, cloud resources, and pricing strategies selected using our BLP and MIP models. Thus, based on cloud providers' APIs, it gives the real cost of BP deployment.

3.5 Conclusion

In this chapter, we gave an overview of our proposed approach for correct cloud resource allocation that provides the optimal deployment cost of time-aware BP. First, we presented our motivation and related problems. Second, a real use case from France Telecom Orange labs is presented as a case study. Then, we presented the different contributions composing our approach. Next chapter is devoted to give more details about existing works related to our research field.

State of the Art

Contents

4.1 Introduction	61
4.2 Temporal constraints and Cloud resources in BP	62
4.2.1 Formal specification in BP	62
4.2.2 Formal verification	69
4.3 Optimization of resource allocation	78
4.3.1 Resource allocation	80
4.3.2 Cloud resource allocation	83
4.3.3 Pricing strategies	90
4.3.4 Synthesis	91
4.4 Conclusion	93

4.1 Introduction

In Chapter 1, we introduced our research problem concerning the verification and optimization of time-aware cloud resource allocation in BPs. In this Chapter, we review the literature for the relevant state-of-the-art works corresponding to our research problem. The latter are: (1) how to formally specify cloud resources' temporal constraints in BP models? (see Section 4.2.1), (2) how to formally verify the temporal correctness of cloud resource allocation in timed-aware BPs? (see Section 4.2.2), and (3) how to optimize the deployment cost of BPs in cloud resources? (see Section 4.3). It helps us to position our work among other works in the literature and assists us to justify our problem statement by comparing and contrasting the existing works with our contribution. Finally, in Section 4.4, we conclude this Chapter. Furthermore, at the end of each section, we provide a synthesis comparing the current research works and their shortcomings with our work, which help us to clearly position and motivate this thesis work.

4.2 Temporal constraints and Cloud resources in BP

In the literature, several existing works have viewed the domain of BPM from different perspectives, i.e., the control-flow perspective, the data perspective, the artifact perspective, or the resource perspective. Among them, the control flow perspective has been widely researched as it helps to streamline the temporal aspect of the BPs, which is important for timely execution of the processes that are based on certain Service Level Agreement (SLA). Further, to be cost-effective and gain competitive advantage, it is critical for an organization to properly manage its resources, i.e., both humans and machines/non-human (automation intensive processes). While the (cloud) resources perspective in BPM has been an area of interest for quite some time, most of the work has been focused on the cloud resource properties such as shareability, elasticity [27, 33]. Further, cloud resources are considered always available. In other words, there is a lack of work done on the modeling and management of cloud resources with limited temporal availabilities in the BPM lifecycle. Thus, this thesis project is motivated towards handling the issues related to management of cloud resources' temporal availabilities in the BPM domain.

4.2.1 Formal specification in BP

This section presents some existing works related to formal specification of activity temporal constraints (Section 4.2.1.1) and resource perspective such as human, machines, and cloud resources in BPM context (see Section 4.2.1.2).

4.2.1.1 Activity temporal constraints specification

Various researchers have tackled the temporal dimension in BP [7, 29–31, 146–150]. For instance, Arévalo et al. [147] suggested a model-driven approach to extend BPMN metamodel in order to support time perspective. Thus, they defined a time rule taxonomy in most BPs that can be applied with current BPMN 2.0 standard in a declarative way.

D. Gagné et al. [29] extended BPMN with temporal dimension. Nevertheless, they did not propose a formal specification. Moreover, authors in [30, 31] have concentrated on the issue of the formal specification of temporal constraints related to BP activities: relative and absolute constraints. Those authors proposed not only a formal specification but also a set of time decorators in BPMN.

Besides, Lanz et al. [35] suggested 10 time patterns to foster the comparison of existing PAISs with respect to their ability to deal with temporal aspects. Figure 4.1 summarizes the proposed time patterns. As shown in Figure 4.1, authors identified 4 distinct categories based on their semantics. Particularly, the time patterns constitute solutions for representing commonly occurring temporal constraints in PAISs. Pattern Category I (Durations and Time Lags) provides support for expressing durations of different process granularities (i.e., activities, activity sets, processes, or sets of process

instances) as well as time lags between activities or – more generally – between process events (e.g. milestones). Pattern Category II (Restricting Execution Times) allows specifying constraints regarding possible execution times of single activities or entire processes (e.g., activity deadlines). Category III (Variability) provides support for expressing time-based variability during process execution (e.g., varying control-flow depending on temporal aspects). Finally, Category IV (Recurrent Process Elements) comprises patterns for expressing temporal constraints in connection with recurrent activities or process fragments (e.g., cyclic flows and periodicity) [35]. Figure 4.2 presents an example of a BP extended with time patterns. Then, Lanz et al. [146] proposed a formal semantics of time patterns in order to avoid ambiguities and to ease their use as well as their implementation. Further, to enable pattern use in a wide range of process modeling languages and pattern integration with existing PAISs, this semantics is expressed independently from any particular process meta model. Altogether, the presented pattern formalization will foster the integration of the temporal perspective in PAISs. Nevertheless, in our work, we consider temporal constraints not only for BP activities but also for cloud resources. Moreover, we propose an approach to formally verify the temporal correctness of cloud resource allocation in a time-aware BP.

Pattern Catalogue
Category I: Durations and Time Lags
TP1: Time Lags between two Activities
TP2: Durations
TP3: Time Lags between Arbitrary Events
Category II: Restricting Execution Times
TP4: Fixed Date Elements
TP5: Schedule Restricted Elements
TP6: Time-based Restrictions
TP7: Validity Period
Category III: Variability
TP8: Time-dependent Variability
Category IV: Recurrent Process Elements
TP9: Cyclic Elements
TP10: Periodicity

Figure 4.1: Time patterns proposed in [7]

Time modeling and management in the clinical workflow domain has been widely investigated by Combi et al. [148–150]. In [148], the authors proposed a general conceptual workflow model considering both activities and their temporal constraints. Among the proposed temporal constructs, we can notice: the duration (activity duration) and delays (edge duration), the relative constraints, the absolute constraints,

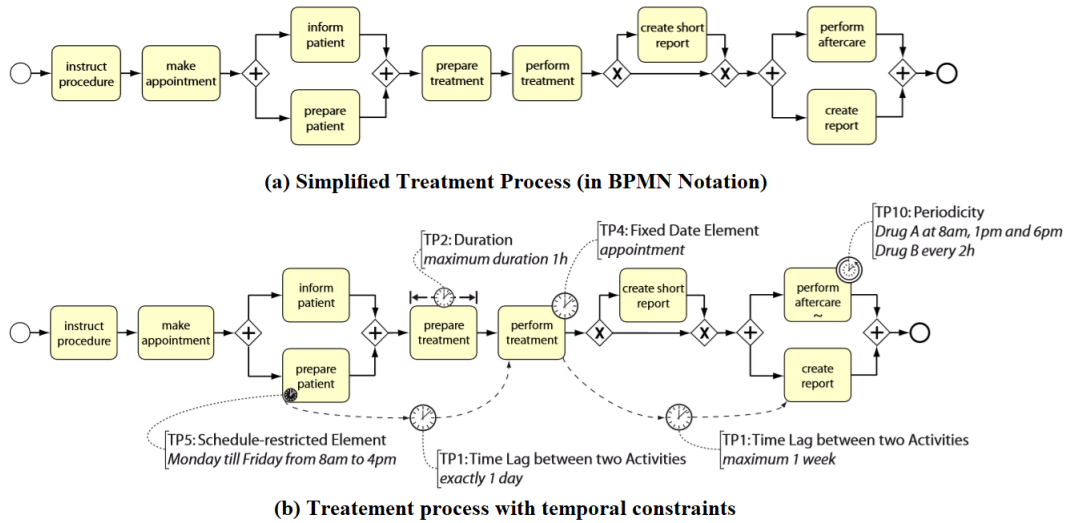


Figure 4.2: Treatment process [7]

and the periodic constraints. Based on these constructs, the authors developed a tool named Temporal Workflow Analyzer (TWA) to support workflow modeling at workflow design time.

However such approaches, working on the specification of activities temporal constraints, lack considering resource perspective. Therefore, to cope with this research gap our work is motivated to consider resource, more specifically cloud resources, and to ensure their specification enriched with temporal availabilities in time-aware BP.

4.2.1.2 Resource perspective specification

In the literature, various authors focused on the specification of resource perspective in BPM such as [8, 38–42]. For example, Cabanillas et al. [8, 38, 39] represented the importance of resource management in BPs and the need for evolution of PAIS into Process and Resource-Aware Information System (PRAIS). More precisely, Cabanillas et al. [8] proposed RALph (Resource Assignment Language graph), which is a graphical notation for human resource assignments in BPs that derives its formal semantics from Resource Assignment Language (RAL) [151]. As shown in Figure 4.3, the RALph notation proposes four types of resource entities: Organizational Unit, Positions, Person, and Roles. Next, it defines the Capability entities, which are persons having a specific capability. It has several Connectors (similar to control flow connectors), which are used to express the resource assignment. In RALph, the authors also considered the resource dependencies and the resource-activity dependency. Figure 4.4 illustrates a process of patient examination (Figure 4.4(a)) and the application of the RALph language to the patient examination process (Figure 4.4(b)). More-

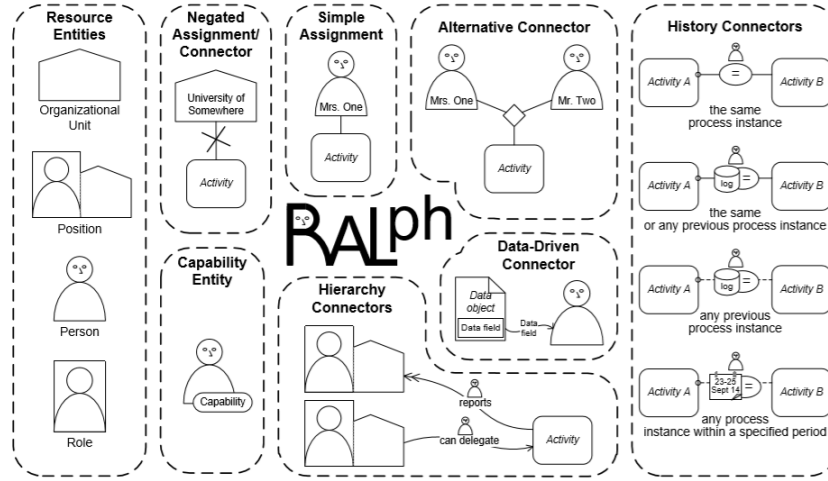
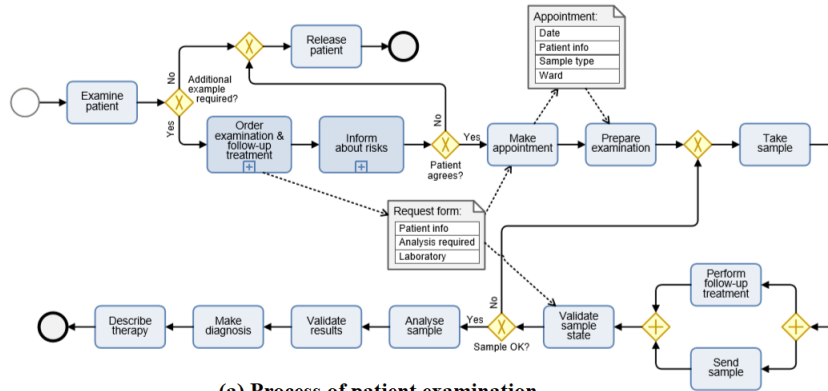


Figure 4.3: The Ralph language [8]

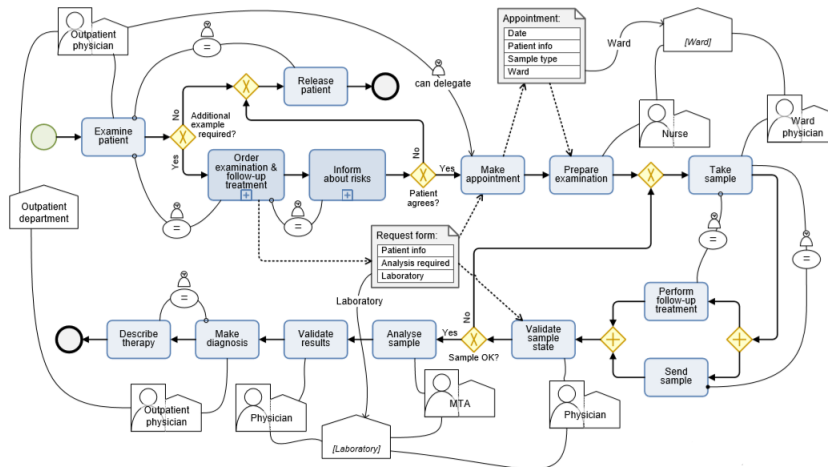
over, in [40], authors extended BPMN 2.0 meta-model to support the modeling and visualization of resource perspective requirements. It considers three aspects of the resource perspective: resource structure, authorization, and work distribution. We notice that the authors' main focus is on human as well as their behavior. However, they do not neither enable cloud resources' representation nor consider verification. Besides, an explicit support of cloud resource temporal availability is missing. In contrast, our work integrates cloud resources and their pricing strategies and seeks for checking the temporal correctness of cloud resource resource allocation.

Russel et al. [41] proposed the Workflow Resource Patterns (WRPs) to support the representation of a resource along with its utilization in a workflow. They grouped these patterns into different categories such as creation patterns, push patterns, pull patterns, detour patterns, etc. For instance, in the creation pattern, they have a Role-Based Allocation pattern (Pattern R-RBA), which helps to specify constraints (at design time) wherein only a resource belonging to a specific role can execute a specific task. In another work, Russel et al. [42] evaluated the BPEL4People and WS-HumanTask extensions to WS-BPEL 2.0 based on the WRPs.

All the aforementioned works consider resources such as human and machines but neglect cloud resources. In the following, we present some existing works dealing with cloud resource management [27, 33, 152, 153]. For example, Hachicha et al. [33] proposed an extension to BPMN meta-model to optimally manage resources deployed in the cloud through resource constraints verification. They aim to formalize the consumed cloud resources using a shared knowledge base. Therefore, authors proposed a semantic framework for resource-aware BP development in the cloud. The same authors in [152] proposed process configuration concepts for cloud computing in BPMN. More precisely, they defined an approach for modeling configurable pro-



(a) Process of patient examination



(b) Process of patient examination with RAL

Figure 4.4: Process of patient examination [8]

cesses with configurable cloud resource allocation operators. In this manner, the BP’s designer explicitly model resource allocation alternatives in multi-tenant process models including elasticity and shareability. However, temporal perspective for cloud resources is not studied.

Few authors suggested *formal* specification of cloud resources in BPMN. Ben Fraj et al. [153] presented a model-driven approach for the specification and the execution of cloud workflow applications composed from cloud services. The approach shows also how the proposed modelling constructs are applied to model and represent a flexible workflow from cloud services specified by BPMN to its running platform by BPEL4WS. Boubaker et al. [27], intended to offer a formal definition of the resource perspective in BPs to ensure correct and optimal cloud resource allocation in BP modeling. Literally, this work proposes a formalism based on Event-B to specify

cloud resource allocation policies in BP models. The model, in this work, considers different cloud properties such as elasticity and shareability. Though, authors [27] do not consider the temporal constraints.

4.2.1.3 Time and resource specification

Several research works focused on the specification of both: *time* and *resource* perspectives in BPM domain [154–157]. For example, Wang et al. [155] proposed a modeling approach for a kind of workflow constrained by resources and non determined time using Petri net. Watahiki et al. [154] extended BPMN to handle temporal, concurrency and resource constraints. However, the scope of this paper is limited to a small subset of BPMN elements. Added to that, the extension proposed in this work grants specifying temporal constraints related to only one activity within the BP model and does not consider advanced temporal constraints related to a set of activities like temporal dependency. Ogata et al. [156] described how to specify Formal Time and Resource-Sensitive simple BP (Formal TR-SBP) in Maude [158] and Alloy [159] specification languages. Whereas, Cheikhrouhou et al. [157] proposed a formal specification of time-aware BPs and their allocated cloud resources using Timed Petri Nets (TPN).

4.2.1.4 Synthesis

Table 4.1 summarizes existing approaches, presented above, discussing the specification of process perspectives such as time, resource, etc. The approaches are classified based on criteria relevant to our work: (1) control-flow perspective, (2) activities temporal constraints, (3) resource, (4) formal, and (5) modeling language. In Table 4.1, we use the following nomenclature: "+" to depict that the approaches satisfy the corresponding criteria, "-" to depict that they do not satisfy the corresponding criteria. Overall, these approaches take into account the control-flow perspective, wherein some of them consider activity temporal constraints and/or resource perspective in BPs and propose formal specifications. These approaches extend the concepts from the BPM domain to include the definition and representation of human/machine resource behavior [8, 38–40, 154, 155, 160–162], which is needed to optimally manage the human/machines resources involved in the processes. There is a need for such approaches as domain specific processes (e.g. health care) involving human/machines are scarce and costly. However, our work mainly focused on modeling and formally specifying cloud resources (and their temporal availabilities) into BPs. The current approaches in the literature from the context of inclusion of cloud computing domain in the BPM domain were discussed in detail in Section 4.2.1.2 of this chapter. Most of the approaches [32, 33] extended the BPs with cloud resources while considering elasticity and shareability properties. Others [152] integrate the cloud resource perspective in configurable BPs and offer a graphical tool to model various resource

Table 4.1: Summary of the literature study of perspectives specification in BP

Work	Control-flow perspective	Activities temporal constraints	Resource			Formal	Modeling language
			Human	Machine	Cloud		
[29, 147]	+	+	-	-	-	-	BPMN
[7, 30, 31, 146]	+	+	-	-	-	+	BPMN
[8, 38, 39]	+	-	+	-	-	+	Ralph
[40]	+	-	+	-	-	-	BPMN
[154]	+	+	+	+	-	-	BPMN
[155, 160-162]	+	+	+	+	-	+	Petri nets
[33, 152, 153]	+	-	-	-	+	-	BPMN
[32]	+	-	-	-	+	+	BPMN
[157]	+	+	-	-	+	+	Petri Nets
Our approach	+	+	-	-	+	+	BPMN

allocation alternatives in multi-tenant process models. But, the proposed approaches failed to address cloud resources constraints such as temporal availabilities which has been only studied in [157]. Finally, we observe from Table 4.1 that BPMN is the most used modeling language.

Unlike the researches detailed above, in this thesis, we formalize cloud resources and integrate pricing strategies properties such as temporal constraints with BP concepts such as activities temporal constraints. First, we defined pricing strategies proposed by cloud providers and specified based on temporal constraints as detailed in Section 2.4.2. The formal model proposed in this thesis work is used to specify the cloud resources and BPM concepts in the process models developed graphically. Further, we propose a BPMN extension to support our proposed formal model which can help a BP designer to model a time-aware BP deployed in cloud resources offering different pricing strategies.

4.2.2 Formal verification

In this Section, we give an overview of *formal* verification methods used in BPM context. For that, we present, first, formal verification methods (Section 4.2.2.1) and existing works related to three aspects of process models verification: structural (Section 4.2.2.2), temporal (Section 4.2.2.3), and resource (Section 4.2.2.4).

4.2.2.1 Formal verification methods

The process verification is the task of determining and checking whether it exhibits erroneous behaviors. This refers to process correctness checking. Hence, the verification could be applied at design time in order to detect possible errors, and if so, the model should be modified before execution. The validation aims at checking whether the system actually behaves as expected or not. The later is context dependent and can only be applied with knowledge of the intended BP. Since PAIS rely on process models for organization's work execution, careful verification and validation of process models at design time can greatly improve the reliability and efficiency of such systems. Therefore, there have been many efforts to achieve that by defining formal semantics of process models and applied various logics and formal methods [16]. The most widely used process modeling languages do not have formal semantics, notably EPC [22], BPMN [21], UML activity diagram [24]. Therefore, these modeling techniques need to be mapped into formal models in order to be verified, e.g., [163–165].

Many existing approaches used simple languages such as workflow graph model [166]. Further, Petri Net formalism was widely explored due to its understandable and graphical notation as well as well-defined semantics. More specifically, the Workflow nets [162, 167] are its most notably sub-class, that were used as intermediate formalism to verify and analyze workflows/BPs, e.g., [76, 168, 169]. In the literature, some authors [170, 171] relied on process algebra (i.e., π calculus [172]) and event calculus [173] to formally verify BPs. Model checking techniques were used in [174, 175] to show BPs' consistency through the automatic verification of basic requirements such

as the termination and reachability of states. It is noteworthy that timed automata, due to its formal syntax and semantics, has been largely adopted to formally model BPs that will be analyzed using model checking tools [176–179]. For this reason, we use in our work timed automata to formally specify and verify that a cloud resource allocation is temporally correct.

4.2.2.2 Structural verification

The structure of a process model defines activities' order in a model, and the objective of verifying this structure is to verify the structural correctness of a process (workflow) specification. Such verification is usually based on an extension of a kind of formal method, for example, directed graph, temporal logic, or Petri Net [180]. For instance, Sadiq and Orłowska [9] employed directed graphs to specify a process and to verify its structural correctness through analyzing this directed graph. Some structural conflicts, such as deadlock (Figure 4.5(c)) or lack of synchronization, can be identified in the process specification [180]. Indeed, joining exclusive choice paths with a synchronizer results into a deadlock conflict. A deadlock at a synchronizer structure blocks the continuation of a workflow path since one or more of the preceding transitions of the synchronizer are not triggered.

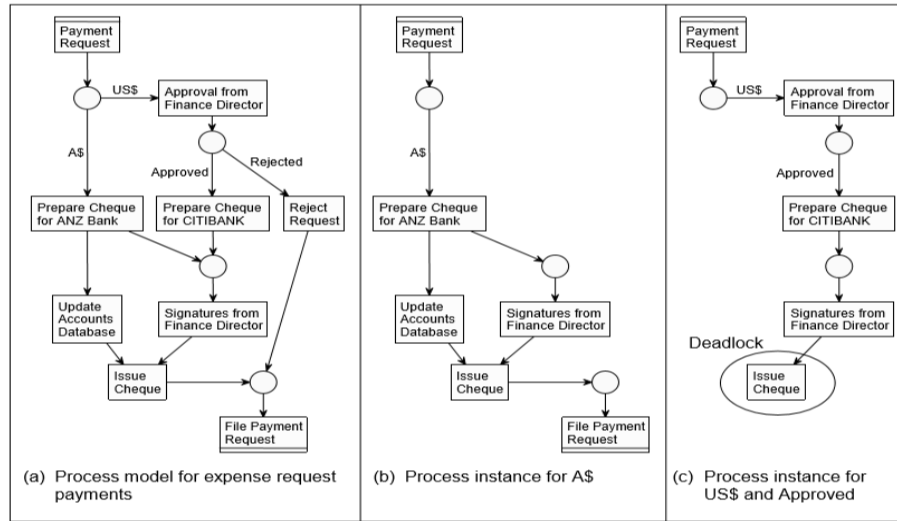


Figure 4.5: Example process model with deadlock structural conflict [9]

Besides, Greco et al. [181] relied on temporal logic to describe workflows mapped onto logic expressions which are analyzed to verify the structural correctness of the specification. Van der Aalst [182], Van der Aalst and Hofstede [183], and Lee and Lai [184] used Petri Net to map processes in order to verify the structural correctness of a process specification.

We notice, from the presented proposals, that authors focus on structural verification in BPM context. However, they do neither verify temporal consistency nor resource allocation correctness. In contrast, our work deal with the temporal verification of resource allocation in time-aware BPs.

4.2.2.3 Temporal verification

Dealing with time and time constraints is crucial in designing and managing business and science processes [180]. As a result, time management should be considered as a fundamental part of BPM systems [185, 186]. Various research works have tackled the issue of temporal constraints verification such as [30, 73, 134, 148–150, 187–198]. For example, Eder et al. [187] specified a timed workflow graph by defining the earliest start time and latest end time for each activity node.

Marjonvic et al. [188] defined a temporal duration for each individual activity and proposed verification algorithms to check the temporal requirements and inconsistencies. In another work, Marjonvic et al. [189] suggested a graphical and a formal model for absolute temporal constraints. Further, they proposed a dynamic verification mechanism. Zhuge et al. [190] introduced an approach to verify the consistency of temporal constraints at build time and runtime in BPM context. Bettini et al. [194] focused on temporal workflow models and used Temporal Constraint with Granularity (TCG) to model the control flow and temporal constraints. Also, in this work, authors proposed temporal constraints reasoning and management tool that check the consistency of complex temporal requirements.

Various authors utilized *formal* methods to check process temporal correctness. Cheikhrouhou et al. [30, 73] proposed a set of rules to transform BPs modeled in BPMN language and enriched with relative and absolute temporal constraints into timed automata models. Figure 4.6(b) depicts the timed automata resulting from the application of the authors' algorithm, named *From Timed Process to Timed Automata*, for the Shipper process (Figure 4.6(a)). Then, using the model checking tools (UPPAAL), authors verified the satisfaction of some CTL properties in order to verify the temporal consistency of BPs. Morales et al. [196] presented an approach to transform BPMN models into timed automata models based on model checking verification mechanism. Zhao et al. [199] focused on specification and verification of the temporal constraints of the service-based BP model based on timed automata models.

Adam et al. [192] used Petri nets to specify temporal constraints in workflow and check the temporal feasibility at build-time. Further, Huai et al. [134] verified the BPMN models using timed Petri Nets. Basically, they analyzed the process structure such as deadlock and tested if there is a conflict between the model temporal constraints. To do that, they defined the Petri net reachability graph. They used also a time choreography verification algorithm to check time conflicts. Besides, Wong et al. [193] presented an approach that uses a timed semantic function which

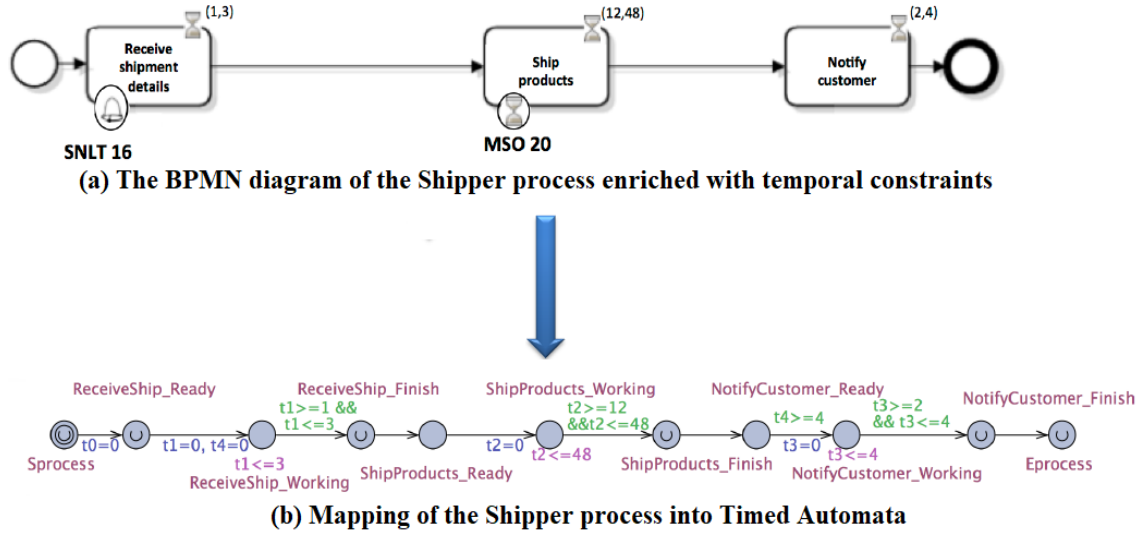


Figure 4.6: BP model mapped into BPMN model [10]

takes as input a diagram describing a collaboration. Then, the function provides a communicating sequential processes modeling the time behavior of the diagram. Using the Failures-Divergence Refinement [200] system, some properties are verified.

Yang et al. [195] proposed to model timed service BP using Petri net. More precisely, they developed a framework to model a service BP composed of a service model extended with temporal specifications. Next, they generated automatically timed properties in the form of temporal logic formulae to verify the temporal correctness. Additionally, taking Petri nets as models, Ling and Schmidt [201] provided a time interval extension of workflow nets to support the modeling and analyzing of time constraints in workflow systems. They putted more emphasis on checking the soundness of workflow process definitions, but they considered very few time constraints.

However such approaches, addressing the verification of activities temporal constraints, lack considering resource perspective. Thus, to address this research gap our work is motivated to integrate resources, more specifically cloud resources, and to guarantee the resource allocation correctness in time-aware BPs.

4.2.2.4 Resource allocation verification

Resource management has been widely investigated [202–205]. Indeed, authors introduced different extensions of workflow nets to take into account resources. Besides, they studied also the problem of soundness criteria to prove the decidability of workflow nets. For instance, Bashkin et al. [202] modeled workflow using Resource-Constrained Workflow Nets (RCWFnets) and presented an algorithm that computes

minimal sound resource. Whereas, in [203,204], authors defined a more general class of RCWFnets considering, first, that resources are available initially and after terminating all cases. Second, they also require that for any reachable marking, the number of available resources does not override the number of initially available resources [202].

Human resources are crucial in BPM as they are responsible for process execution or supervision [206]. That is why, there are several works that manage correctly human resource allocation [11, 12, 161, 207] For instance, Li et al. [161] presented an analysis approach to verify the human resource consistency of workflow specification. Wang et al. [207] proposed a formal approach for the modelling and analysis of emergency response processes, which has taken human resources into consideration.

Other authors focus not only on the specification but also on the verification of human resources in BPM context [11, 12]. Netjes et al. [11] developed a model for the analysis of resource-constrained processes and the process alternatives. They proposed an abstraction of a BP consisting of a process structure formed by activities in sequence or parallel, a generic resource module and a method to allocate resources. For the modeling and analysis of resource-constrained processes, Netjes et al. [11] used Colored Petri Net (CPN) [208] tools that provides support for the construction, simulation and performance analysis of high-level Petri Nets [11]. Figure 4.7 shows an example process that is modeled using the developed CPN model. This example process deals with the opening of a bank account. When a registration for a bank account is received the data is entered into the system. Secondly, the bank account is initiated and finally a letter is sent to the customer. The process model is built with three task building blocks and the resource module. Besides, Shin et al. [12] proposed a static approach for resource analysis. As can be seen in Figure 4.8, the approach extends the Trace Flow Graph (TFG) Translator and Resource Constraint FSM Translator components to generate the inputs to the Finite State Verifier (FLAVERS). The TFG extensions have the effect of augmenting this graph with the representations of resource allocation events and resource utilization policies in addition to the TFG's existing specifications of activities and artifact flows. Augmentations to the Resource Constraint FSM Translator convert the various resource policy specifications into constraint FSMs that are used to exclude infeasible resource allocation behaviors from the analysis search space.

There exist previous researchers that consider cloud services such as [209–212]. Indeed, Klai and Ochi [211] used Symbolic Observation Graphs (SOG) to abstract cloud services and check the correction of their composition with respect to event- and state-based LTL formulae (Hybrid LTL). Rezaee et al. [212] proposed the Fuzzy Inference Cloud Service (FICS) modeled using the CSP process algebra and introduced four formal verification tests to allow strict analysis of certain behavioral disciplines in the FICS. Zhou et al. [209] proposed a method for modeling and verification of resource provisioning as a service in the cloud. To this end, they presented the framework of resource provisioning as a service and the behaviors of its participants. Then, they resorted to UPPAAL to model client, service manager (including alloca-

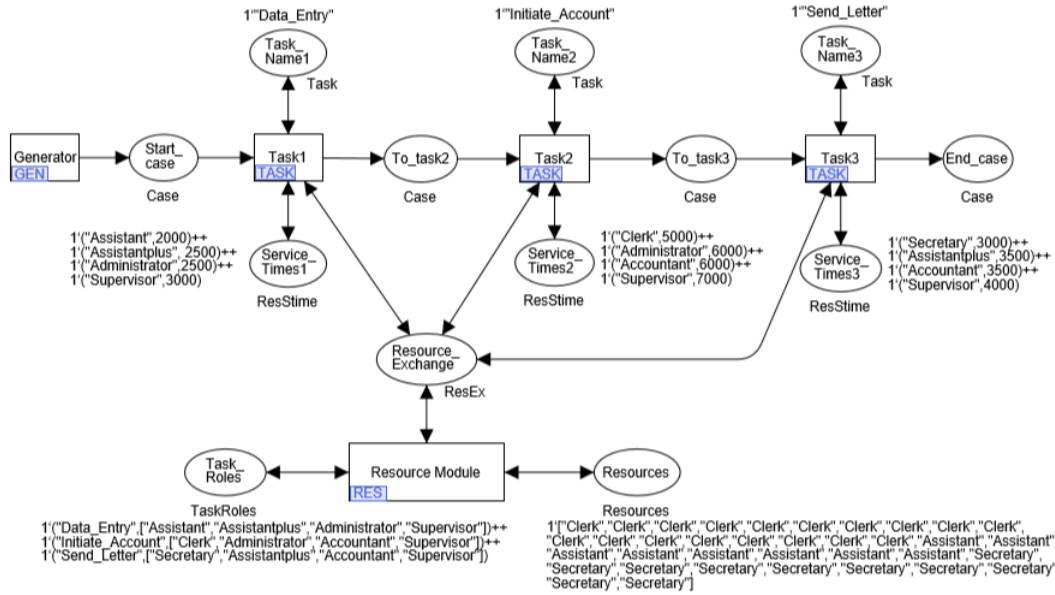


Figure 4.7: Process model: Opening a bank account [11]

tor, finish monitor, and time monitor), and resource service, respectively. Finally, Zhou et al. [209] defined some consistency properties that a service scenario needs to satisfy and formally verify. As a result, they can check the satisfaction of this properties using UPPAAL model checker. The aforementioned approaches proposed formal models of cloud services that present a good basis for further analysis and verification, contributing to the improvement of security and quality of service.

Several studies have been carried out on the *formal* modeling and analysis of cloud properties such as elasticity [210, 213, 214]. Gambi et al. [214] adopted the state transition systems to formalize cloud-based systems while taking into account elasticity properties. This formalization is then used to automatically generate load test cases focusing on elastic behavior of elastic systems. Elasticity is verified by ensuring that, for each scaling up, there should correspond a scaling down. Berasni et al. [210] adopted a temporal logic called CLTLt(D) (Timed Constraint LTL) to formalize the elastic behavior of cloud-based systems. They formalized properties related to horizontal elasticity, resource management, and quality of service QoS. Then, they proposed to check whether these properties hold or not during execution of a cloud-based system. Authors in [213] used bigraphical reactive systems (BRS) for specifying both structural and behavioral aspects of elastic cloud-based systems. Then, they used Maude's model-checking invariants technique to simulate and verify the elasticity property by ensuring that the cloud system scale up/down when needed. While, the aforementioned approaches [210, 213, 214] considers the elasticity properties

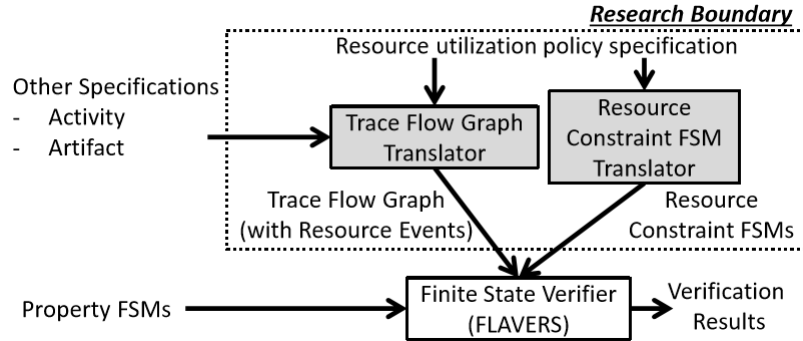


Figure 4.8: Static resource analysis approach overview [12]

of resources or services in cloud-based systems, they do not take into account the process perspective.

Some recent works tackled the problem of cloud resource allocation in BPs [27, 32, 34, 215]. Boubaker et al. [27], using the Event-B formal specification, verified the consistency of cloud resource allocation for process modeling at design time. Besides, they checked the allocation correctness according to users' demands and resource properties. Likewise, the same authors in [32, 215] verified the correctness of cloud resource allocation in BPs considering properties such as elasticity and shareability. Garfatta et al. [34] rely on Coloured Petri net formalism to model formally the cloud resource perspective in BP taken into consideration their properties such as vertical/horizontal elasticity. So, the BP designer can model correct resource allocations in BPs to avoid runtime errors.

Even so, we observe that the main focus on resource allocation verification take into account various resource constraints such as number, availability, concurrency, etc. Moreover, approaches addressing the verification of cloud resource allocation do not consider temporal availabilities constraints. In contrast, our work extends cloud resources with temporal perspective and verifies the temporal correctness of cloud resource allocation in BPM context.

4.2.2.5 Time and resource allocation verification

Many attempts have been made [132, 154, 155, 216–219] with the purpose of ensuring a correct resource allocation in a timed-constrained workflow. Hsu et al. [216] constructed an incremental methodology that aims to analyze the resource consistency and temporal constraints after each edit unit defined on a workflow specification. The methodology introduces several algorithms for general and temporal analyses. The output returned right away can improve the judgment and thus the speed and quality on designing. Arévalo et al. [219] proposed an MDE-based approach to generate BPMN models from legacy information systems considering time constraints and

resource allocation rules.

Some authors resorted to *formal* methods to check the correctness of workflow/BP under temporal and resource constraints [132, 154, 155, 217, 218]. For instance, Wang et al. [155] presented an approach for the analysis of concurrent workflows under resource constraints and non determined time based on Petri nets. With the obtained R/NT_WF_Net model and its reachability graph, they analyzed the timing factors influencing the implementation of the whole workflow, presented the methods to verify the risks of all implementation cases, and founded the best implementation case for the workflow. Zeng et al. [218] proposed a resource conflict detection approach and removal strategy for emergency response processes constrained by resources and time. Based on the RT_ERP_Net, the earliest time to start each activity and the ideal execution time of the process can be obtained. Authors utilized the conflict detection algorithms and a priority-activity-first resolution strategy to detect and remove the resource conflicts in the process. In this way, real execution time for each activity is obtained and a conflict-free RT_ERP_Net is constructed by adding virtual activities [218]. Zhong et al. [217] discussed the analysis on resource constraints of concurrent workflows. For that, they defined the Time Constraint Workflow Net and mapped the workflow concepts onto this net to model workflow. Then, Zhong et al. [217] identified the problem of resource constraints in WfMS. After that, they proposed corresponding analysis method with pseudocode algorithm to check the resource consistency for concurrent workflows. Furthermore, they suggested several ways to remove potential resource conflicts from workflow model.

Watahiki et al. [154] proposed an extension to BPMN in order to support temporal, concurrency, and resource constraints. Moreover, they generated automatically from the extended BPMN models, timed automata models based on their mapping rules. The aim of the approach is to check deadlocks and bottlenecks. Besides, based on a sprouting graph of TWF-nets, Du et al. [132] have presented a dynamic checking approach of temporal constraints for concurrent workflow processes with resource constraints. They used UPPAAL to verify the correctness of the proposed approach, and they also used a concrete example to prove the usability and scalability of the approach.

We mention that several works deal with formal verification of cloud resource allocation in timed-constrained BPs [157, 220–225]. Massive parallel business workflows running in the cloud are prone to temporal violations (namely intermediate runtime delays) due to various reasons such as service performance fluctuation and resource conflicts [224]. Thus, authors in [224] presented a propagation-aware temporal verification strategy for parallel business cloud workflows. More precisely, they proposed to analyze the effect of time delay propagation in cloud workflow systems. Then, they presented a novel temporal verification strategy based on a new propagation-aware throughput consistency model which includes the propagation effect. Next, Luo et al. [223] developed the idea of "adaptiveness" in their design strategy in order to detect temporal violations and to achieve on-time completion of time-aware business cloud

workflows. So, they presented an adaptive temporal checkpoint selection strategy. Besides, they proposed a strategy to handle the temporal violation. This strategy can determine the required lifecycle of cloud services. Besides, Wang et al. [225] proposed a new sliding-window based checkpoint selection strategy for detecting temporal violations. Indeed, the strategy selects the next observation time interval based on the overall temporal consistency state at last checkpoint. The method will stay fixed observation time interval for next checkpoint when a temporal insistence state is detected. Otherwise, next observation time interval along the execution timeline will be enlarged. All the aforementioned works presented focus on temporal violations for parallel business workflows running in cloud caused by various reasons such as service performance fluctuation and resource conflicts. However, they did not consider that cloud resources have limited temporal availabilities.

Whereas, Cheikhrouhou et al. [157] proposed an approach that aims at assisting BP designers to identify necessary cloud resources with respect to temporal and financial restrictions on BPs. The former minimizes the search time for cloud resources while the latter minimizes the cost of leasing these resources. For that, Cheikhrouhou et al. [157] proposed a formal specification of such BPs and their allocated cloud resources using Timed Petri Nets (TPN). They provided, also, an automatic transformation of cloud resources, according to their pricing strategies, into Timed Petri Nets. Finally, they proposed a formal verification step that checks BP correctness along with meeting deadlines. The authors' approach was inspired by our contribution. Specifically, Cheikhrouhou et al. [157] followed our steps to verify the temporal correctness of a cloud resource allocation in a time-aware BP. However, they used TPN in order to check other properties.

4.2.2.6 Synthesis

Table 4.2 lists the approaches, detailed in Section 4.2.2, focusing on formal verification in BPM domain. We classify those works based on different criteria such as: (1) control-flow perspective, (2) activities temporal constraints, (3) resources, (4) resources temporal constraints, (4) formal language, and (6) Technique. In Table 4.2, we use the following nomenclature: “+” to depict that the approaches satisfy the corresponding criteria, “-” to depict that they do not satisfy the corresponding criteria, “+/-” to depict that the approaches partially satisfy the corresponding criteria.

As reported in Section 4.2.2, most of the approaches detailed above focused on control-flow perspective. Time perspective is partially handled by some approaches as visible in Table 4.2. Moreover, resource perspective has gaining much attention while they are crucial in BPM and responsible for process execution. For instance, human and machine resources have been widely investigated [11, 132, 154, 154, 155, 161, 188, 190, 216, 217, 226]. Most of the authors studied the verification of resource allocation in BPs under various resource constraints such as number, availability, concurrency, etc. Further, few studies have been published considering not only the verification

of resource allocation correctness but also the verification of temporal consistency of constrained BPs [132, 154, 155, 217, 218]. But, in the current state-of-the-art, there is a lack of work tackling the cloud resource temporal availability for time-aware BPs. Thus, to deal with this research gap it is necessary to consider temporal constraints of both: activities and cloud resources to ensure a correct resource allocation in a time-aware BPs. The verification step was performed by various techniques (formal/not formal) such as algorithms, timed automata, Petri nets, etc. Most of studies rely on formal techniques in order to ensure a correct execution of processes under various constraints such as time, resource, etc.

Contrary and/on complementary to above, our work is focused on formally verifying the matching between cloud resources and activities temporal constraints in BPs. For instance, we propose to transform time-aware BP, deployed in cloud resources and modeled using BPMN, into a formal model, timed automata. The generated models will be the inputs to a model checking tool to verify the satisfaction of some properties which can help the designer to ensure a temporal correct cloud resource allocation in time-aware BPs.

4.3 Optimization of resource allocation

In literature, several existing works have been suggested in the context of optimizing resource allocation execution cost [44, 52, 53, 58, 227–229], execution time [52, 53, 55, 58, 230], energy consumption [51], etc. Among them, the execution cost has been widely researched as it helps to reduce fees, which is important for organizations to satisfy their clients and maximize their profits. Thus, organisations need to properly manage their resources to be cost-effective and gain competitive advantage, i.e., both humans and machines/non-human (automation intensive tasks).

This section helps us to position our work among existing works in the context of optimizing resource allocation's cost, especially in the context of BPM and cloud domains. While optimizing cloud resources allocation in BPM has been an area of interest for quite some time, most of the work has been focused on assignment and/or scheduling of resources while considering various constraints such as deadline, budget, penalty cost, etc. In other words, there is a lack of work done on the optimization of cloud resources in BPM context while considering advanced temporal constraints, resource availabilities, etc. This is due to the fact that BP have just simple temporal constraints such as duration and cloud resources are considered always available and there is a lack of formal modeling of pricing strategies. So, this thesis project is motivated towards handling the issues related to minimizing the deployment cost of time-aware BP's in cloud resources proposed under various pricing strategies.

To understand the existing gaps, we review the current approaches that deal with resource management and optimization in BPM context, especially cloud resources. For simplicity, we divide them into the following: (1) existing approaches for optimizing allocation of resources in Section 4.3.1, (2) existing approaches for *cloud* resources

Table 4.2: Summary of the literature study of verification in BPs

Work	Control-flow perspective	Activities temporal constraints	Resource			Resource temporal constraints	Formal	Modeling language
			Human	Machines	Cloud			
[188, 190]	+	+	+/-	-	-	-	Algorithms	
[187]	+	+/-	-	-	-	-	Timed graph	
[134, 192, 195, 197, 198, 201]	+	+/-	-	-	-	+	Petri net	
[30, 31, 196]	+	+	-	-	-	+	Timed automata	
[12, 205, 207]	+	+	-	-	-	+	Timed automata	
[154]	+	-	+	-	-	-	Algorithms	
[11, 161]	+	-	+	+	-	+	Petri nets	
[216, 226]	+	+/-	+	-	-	-	Algorithm	
[155, 217]	+	+/-	-	+	-	-	Algorithm	
[132, 154]	+	+/-	+	+	-	-	Timed automata	
[27, 32, 215]	+	-	-	-	+	+	Event-B	
[34]	+	-	-	-	+	+	Coloured Petri net	
[223-225]	+	+/-	-	-	+	-	Algorithm	
[211]	-	-	-	-	+	+	SOG	
[212]	-	-	-	-	+	+	Process algebra	
[209]	-	-	-	-	+	+	Timed automata	
[210]	-	-	-	-	+	+	CLTL	
[213]	-	-	-	-	+	+	Maudes	
[214]	-	-	-	-	+	+	State transition systems	
[157]	+	+	-	-	+	+	Timed Petri Net	
Our approach	+	+	-	-	+	+	BPMN	

allocation in time-aware BPs in Section 4.3.2.

It is important to note that, in the field of resource allocation, the optimization problem has been studied as an assignment, and/or a scheduling issues. Some authors consider that there is no difference between them. Whereas others consider that resource assignment is deciding which of several resource will perform which of several activities. But, activities scheduling is the act of deciding at what time an activity is performed [231]. Thus, we divide the detailed approaches based on problem type: assignment or scheduling.

4.3.1 Resource allocation

This section presents some existing works that concentrated on resource allocation management and optimization in BPM context [43, 44, 227, 228, 232]. Indeed, some of those works proposed approaches to optimally *assign* resources in a cost-effective manner. For instance, the appropriate selection of human resources is critical as various factors such as workload or skills have an impact on work performance [227].

4.3.1.1 Resource assignment

An entropy-based reinforcement learning approach for the optimization of cycle time in BPM with the aim of minimizing resource allocation through workload balancing is suggested in [43]. The idea of entropy work lists is inspired by the fact that similar tasks may have some subtasks in common, i.e., accessing locally close archive files for human resource or loading identical files into ram for machine resource. Therefore, minimizing the entropy of work lists associated with each resource leads to reduce the processing time of rather consecutive similar tasks [43]. Using Hidden Markov Model (HMM), Yang et al. [44] described complicated relationships among employees which are ignored by previous approaches. Further, they proposed an optimal approach named Staff Assignment based on Hidden Markov Models (SAHMM) to allocate the most proficient set of employees for a whole BP based on workflow event logs. Zhao et al. [228] presented a resource allocation method considering the resource coordination, the interval between adjacent activities, and distinguishing turnaround time between different resources from event logs. The experiments show that the proposed method can effectively optimize the resource allocation.

Corominas et al. [233] dealt with the assignment of tasks to the members of the multi-functional staff (each worker is able to perform a given subset of types of tasks) of a work centre, during each period (e.g. 1 h) into which the planning horizon (e.g. 1 shift or 1 week) can be divided. For each type of task to perform, all workers who can perform the task do so at equal worker efficiencies. There are constraints that, if possible, should be respected. The objective is that the percentage of working time dedicated by each worker to each type of task be as close as possible to reference values. The problem is modelled as a sequence of assignments, in which appropriate values for the cost matrix depend on the results of the previous assignments. The

obtained results are satisfactory: the solutions meet the constraints, the scheduled percentages steadily approach reference values and the calculation times are very short. Therefore, the work presented constitutes a potential tool for assigning tasks to multi-functional workers in the service industry.

The work presented in [50] addressed the problem of resource allocation for BPs. To this end, authors proposed an approach that would improve the process structure based on resource allocation requirements such that the overall cost was minimized while meeting the process execution time requirement. The structure of a BP was modeled as a Direct Acyclic Graph (DAG). They applied a basic allocation strategy to minimize the overall cost without considering the time limit. Then, an adjustment strategy was applied to adjust the allocation scheme to ensure that the overall execution time of the process did not exceed the time limit.

We note that most of the aforementioned studies proposed approaches to optimally assign (human/machine) resources to perform activities under a set of constraints such as: time. However, they did not focus on cloud resources consumed to run a time-aware BP. Though, in our work, we propose an assignment method that, under a set of constraints, selects for each BP activity the suitable cloud resource, cloud provider, and the best pricing strategy. In consequence, we provide an optimal deployment cost of a time-aware BP.

4.3.1.2 Resource scheduling

Several studies have been carried out on tasks *scheduling* with the aim of reducing resource allocation cost [13, 14, 135, 136, 234–236]. In one hand, some authors focused on optimizing resource allocation to perform independent activities such as [135, 136, 234, 235]. For instance, Afilal et al. [135] proposed a mixed integer programming model and a key performance indicator based heuristic to find a feasible solution that respect different constraints relative to labor regulations and a constraint relative to multiple sites, balance the workload over employees and minimize overload hours. Al Yakoob et al. [136] modeled the problem of employees allocation to gas stations owned by the Kuwait National Petroleum Corporation (KNPC), as a mixed-integer program. Next, due to the problem complexity, a two-stage approach is proposed, where the first stage assigns employees to stations, and the second stage specifies shifts and off-days for each employee. The two-stage approach provides daily schedules for employees for a given time horizon in a timely fashion, taking into consideration the employees' expressed preferences.

Due to a nursing shortage as well as a misdistribution and poor utilization of nurses in many countries worldwide, policy makers demand that Decision Support Systems (DSS) better allocate scarce human resources. That is why, Gutjahr et al. [234] developed a meta-heuristic (MH) approach to the dynamic assignment of pool nurses to hospital departments in a web-based, flexible assignment system for a regional decision maker. Compared to the traditional nurse scheduling literature, the

authors' non-cyclic approach is also characterized by the flexible setting of problem parameters such as nurses' and hospitals' preferences, number of shift types, length and overlapping of shifts, substitutability of nurses, as well as weights for the components of the cost function under consideration of different working regulations for each of nurse.

The construction of a timetable that satisfies all operational rules and needs in an academic institution, while at the same time fulfills as many of the wishes and requirements of the staff and the students, is an important but extremely difficult task for the staff involved. But, in recent years, changes occur more frequently and patching of what has been developed historically is not always the best policy. Thus, Daskalaki et al. [235] presented a new Integer Programming (IP) formulation of a timetabling problem as it appears in many universities, adding however many features that may be distinct in Engineering Schools of Greek universities.

In the other hand, other authors dealt with optimal scheduling in the context of *BPM* such as [13,14,236]. One of the scheduling problems addressed in the literature is employee scheduling which require novel mathematical models and algorithms to deal with the specific nature and size of the problem. To this end, Ismaili et al. [13] introduced a two-phase approach to ensure an effective scheduling in the case of critical tasks that must be executed by human resources (Figure 4.9). The first phase represents a solution for event priority determination to ensure an effective instance scheduling in BP. This solution is based on the analysis of historical data from past BP execution using unsupervised machine learning algorithms for clustering, in order to manage the priority of several events that launch BP instances. The second phase is about resource allocation. In fact, the problem of scheduling in BPs, has several constraints at the same time such as resource availability and reliability, and time. As this problem is considered as an optimization problem, Ismaili et al. proposed a genetic algorithm to solve it in order to achieve an effective matching between the most critical process instance and the most available human resource.

Havur et al. [14] introduced a novel approach for automatically allocating resources to process activities in a time optimal way that is designed while considering concurrent process instances or loops in BPs. To do that, they represented 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. Figure 4.10(a) illustrates a BP model that specifies the process of publishing a book. The input of the program encoded in ASP is: (i) three different instances i1, i2, i3 of the timed Petri net, whose starting times are defined as $t_{0_{i1}}=0$, $t_{0_{i2}}=6$ and $t_{0_{i3}}=11$, respectively; (ii) the organisational model and optional activity times for resources and roles as shown in Figure 4.10(b), and (iii) role-activity relation. The computed optimal resource allocation is visualised in Figure 4.10(c). 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.

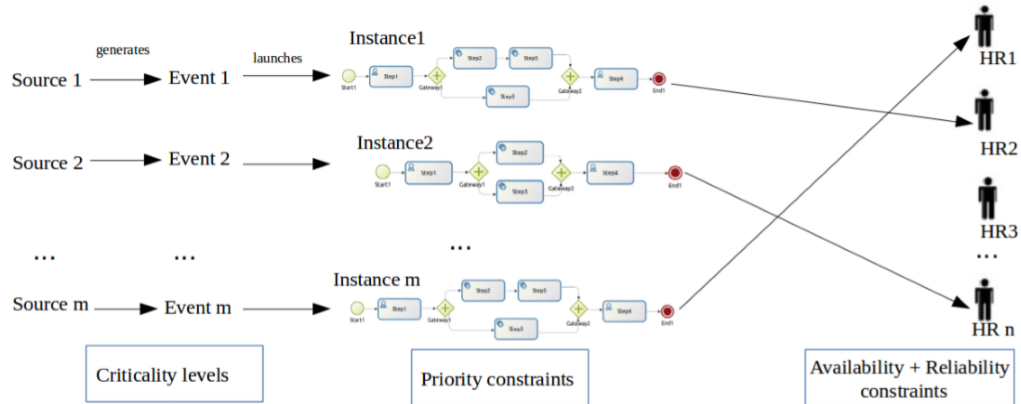


Figure 4.9: Priority-based scheduling of process instances under human resource constraints [13]

Xu et al. [236] investigated how to plan resources for a BP optimally, in aspect of meeting process requirements and rational utilisation of resources. Also, the proposed approach can provide information to organisation for decision making and negotiation with customer in order to better use resources. When a massive number of instances are given, both inter and intra instance dependencies are considered in the planning of each batch of instances to be executed in parallel. As the problem is computationally hard [237], a set of heuristic rules were designed, and two strategies based on these heuristic rules were proposed and compared.

However, such approaches, addressing resource scheduling, lack considering cloud resources which is not helping in addressing the optimization of time-aware BP deployment cost in cloud environment. Furthermore, since we are in a cloud setting, pricing strategies are recommended to be taken into consideration to minimize the organizations spending on IT infrastructure. Hence their use enables significant cost reductions. In our work, we aim at developing a scheduling method, that provides an optimal execution plan for BP activities in order to be optimally executed using cloud resources.

4.3.2 Cloud resource allocation

Due to the shortcomings in the aforementioned approaches to allocate and manage efficiently cloud resources in BPs (see Section 4.3.1), there has been a tremendous growth on the research for integrating cloud resource perspective into the BPM domain [46, 51, 53, 56, 58, 238–241]. Many researchers have put forth the importance of research in the direction of optimizing cloud resource allocation cost in the BPM domain, which is mutually beneficial to both domains. A number of surveys have been proposed to conduct extensive reviews to investigate and analyze the relevant

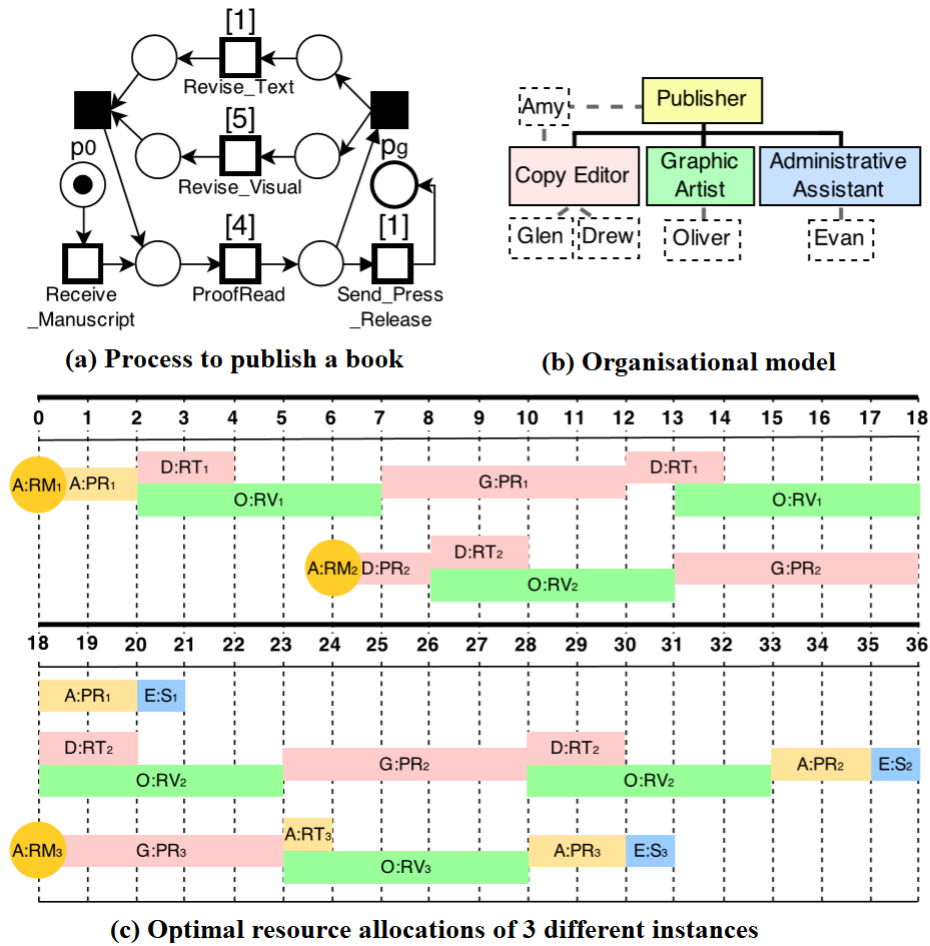


Figure 4.10: Resource allocation of publish a book process [14]

approaches in the context of optimal cloud resource allocation cost, time, energy, etc [238, 240, 242–247]. While optimizing cloud resources cost to deploy BPs (workflows) has been an area of interest for quite some time, most of the work has been focused on assignment and scheduling algorithms to minimize execution cost such as [52, 58, 248–251], and/or task makespan such as [52, 58, 249, 252]. However, most of the authors considered that a single cloud resource has a single hourly cost and tasks have not advanced temporal constraints such as temporal dependency. Thus, this thesis project is motivated towards handling the issues related to the variety of pricing strategies and the complexity of activities' temporal constraints in BPM and cloud computing domains. In other words, it is motivated towards assigning cloud resources to activities or scheduling activities to match cloud resources' temporal constraints while satisfying a set of constraints.

To understand the existing gaps, we review the current approaches that enable the optimization of cloud resource allocation cost in time-aware BPs. For simplicity, we divide them into the following: (1) existing approaches for *assignment* of cloud resources in BPs in Section 4.3.2.1, (2) existing approaches for *scheduling* BP' activities in BPs in Section 4.3.2.2 and (iii) existing approaches for cloud resource allocation in Section 4.3.2.3.

4.3.2.1 Cloud resource assignment

In the literature, a number of works have been suggested for cloud resource assignment [45, 48, 49, 51]. Besides, there is a considerable amount of literature on maximizing cloud *providers* profits such as [47, 253–258]. Some studies have modeled the migration to cloud problem as an optimization problem to find the best deployment of software components on cloud platforms. For instance, Megahed et al. [253] proposed an approach based on an integer linear programming model to optimize the cost a cloud solution, from the service provider's point of view. The aim is to discover the solution with minimum cost that satisfies a set of constraints including client requirements and cloud offering constraints. Moreover, Megahed et al. [254] proposed a novel approach to find the optimal number of instances that are consumed at each time period to answer the incoming queries. Towards this end, they started by learning from the historical behavior of the system in order to predict the probability distributions of the unknown data. Next, they modeled a stochastic program that optimizes the aforementioned trade-off and outputs the optimal provisioning plan. We note that, in this work [254], authors took into consideration the random number of query arrivals that was neglected in the state of the art.

Zhao et al. [258] presented online VM placement algorithms to increase cloud provider's revenue by reducing SLA violation cost. First-Fit and Harmonic algorithms are devised without considering VM migrations, while Least-Reliable-First (LRF) and Decreased-Density-Greedy (DDG) are devised for VM migration considering VM migrations. Ebadifard et al. [47] extended a recent heuristic algorithm called Black hole Optimization (BHO) and presented a multi objective method for workflow application based on Pareto optimizer algorithm. The proposed method considered not only user requirements but also the interests of service providers. Zaman et al. [255], Pandya et al. [256], and Calinescu et al. [259] addressed the issue of dynamic provisioning of VM instances in clouds to generate higher profit by considering various parameters like QoS, cost, time consumption, carbon effect. Besides, Zhang et al. [257] designed a truthful and efficient online VM auctions where cloud users bid for resources into the future for tailor-made VMs with different running durations, targeting social welfare maximization and cloud provider's profit maximization. For that, authors considered server costs in their auction model, and handled the resulting significantly more challenging mechanism design by leveraging a set of latest novel primal-dual online optimization and randomized reduction techniques.

Other authors focused on minimizing cloud *consumers* fees [45, 48, 49, 51, 229, 230, 260, 261]. In one hand, various approaches have been put forward to optimize cloud resources assignment cost to perform independent activities [229, 230, 260, 261]. For example, Han et al. [229] have argued that on-demand scaling of cloud applications raises new challenges for delivering cost efficient services. They proposed a cost-sensitive elastic scaling approach which lowers resource allocation costs by detecting the bottlenecks in a class of multi-tier applications and accordingly scales resources up or down only at these points. Besides, they presented the design and implementation of an intelligent platform based on their scaling approach to achieve cost-effective elasticity.

The objective of Kessaci et al. [230] is to minimize the prices of VM instances and their response time in order to give the best quality of service (QoS) to the clients by reaching their satisfaction rate while providing an interesting profit for the cloud broker. Indeed, they proposed a multi-objective genetic algorithm for cloud brokering (MOGA-CB). It provides a set of Pareto optimal assignments by dispatching the client's virtual machines (VM) requests over the best combination of instances with the minimum cost and response time. This approach uses information provided by the infrastructure service provider (e.g. Amazon) to retrieve the prices of those instances and their different performances to reach its objectives. MOGACB makes this possible by its capacity as a metaheuristic, to explore a wide range of potential solutions to the problem.

Saber et al. [260] defined a multi-objective VM reassignment problem for hybrid and decentralised data centres. They proposed H2-D2, a solution that uses a multi-layer architecture and a metaheuristic algorithm to suggest reassignment solutions that are evaluated by the various hosting departments (according to their preferences). Tordsson et al. [261] proposed an architecture for cloud brokering and multi-cloud VM management. They also described algorithms for optimized placement of applications in multi-cloud environments. The placement model incorporates price and performance, as well as constraints in terms of hardware configuration, load balancing, etc.

In the other hand, several research works [45, 46, 48, 262] focus on optimizing the cost of cloud resources leased to deploy BPs/workflows. For instance, using agent-based systems to simulate processes' enactment, Labba et al. [262] suggested a novel adaptive resource allocation approach for estimating and optimizing the final deployment costs. Besides, the authors extended the Pairwise-Movement Fiduccia-Mattheyses (E-PMFM) partitioning algorithm to deal with the services' QoS changes and to dynamically adapt the initial deployment. The experimental results demonstrate the efficiency of E-PMFM algorithm and show that the deployment costs are sensitive to the initial deployment and the used partitioning algorithm. Rekik et al. [45] defined an approach for generating an optimal process variant deployment into a cloud federation. Given (i) a configurable process model with its set of allowed configurations and non-functional requirements for a specific business context and

(ii) a cloud federation environment with its constraints and resources' properties, the proposed approach provides the process variant having the optimal deployment under the aforementioned constraints. To do so, authors adopted a (0-1) linear programming optimization strategy, with a quadratic objective function (i.e., minimizing the total execution cost which is equivalent to minimizing the resources allocation and the inter/intra resources communication) and the above mentioned disjunctive constraints. Moreover, Rosinsky et al. [46] presented an Integer Linear Programming (ILP) model and a genetic algorithm that aims at finding the best allocation strategy while limiting the number of migrations per tenant.

Fakhfakh et al. [48] proposed an algorithm composed of three stages. In the first stage, the number and the type of VMs associated to each task are determined. The goal is to minimize the execution cost, while meeting a user specified deadline. Then, based first stage results, they applied an adjustment process that reduces the wasted time fractions produced by the assignment step. It consists in consolidating tasks in the same VM instance. Afterwards, they extended their algorithm to take into account the dynamic changes of workflow.

Other authors focused on optimizing other parameters such as energy [51] and security level [49]. Goettelmann et al. [49] suggested an algorithm for deploying BPs on different cloud platforms under security constraints. The main idea was to split a BP into sub-processes and deploy them in different clouds to meet security requirements. A security level was assigned to each BP task describing its security requirements, and arbitrary security levels were assigned to cloud providers. The algorithm would select the suitable cloud providers which offered the minimum communication costs between the derived sub-processes. Whereas, Chen et al. [51] proposed an available budget preassignment method to reduce the energy consumption. In this step, each task selects the combination assignment of VM and frequency that has the minimum energy consumption while satisfying the cost budget of the task.

Contrary to above, our work is concentrated on assigning optimally cloud resources to BP activities under various constraints, from the cloud consumer point of view. To this end, we propose a BLP that assigns for each activity the suitable cloud resource and the best pricing strategy in order to minimize BP's deployment cost while respecting a set of requirements such as time, penalty, RAM, vCPU, etc.

4.3.2.2 Cloud resources scheduling

The task scheduling problem in distributed computing systems is an NP-hard optimization problem which plays an important role in optimizing cloud utilization. It also effects on QoS in the cloud environment by optimizing service cost and service response time [263]. Various authors concentrate on *BP/workflow* scheduling algorithms in cloud environments [15, 46, 52–58]. Besides, a considerable amount of surveys have been proposed to conduct extensive reviews to study and analyze the relevant approaches in the context of scheduling BPs/workflows in cloud environment

such as [238, 240, 247].

Pandey et al. [15] presented a scheduling heuristic based on Particle Swarm Optimization (PSO). They used heuristics to minimize the total cost of execution of application workflows on cloud computing environments. They obtained the total cost of execution by varying the communication cost between resources and the execution cost of compute resources. They compared the results obtained by their heuristic against Best Resource Selection (BRS) heuristic. Pandey et al. [15] founded that PSO based task-resource mapping can achieve at least three times cost savings as compared to BRS based mapping for their application workflow. In addition, PSO balances the load on compute resources by distributing tasks to available resources. Figure 4.11 presents an example of a workflow composed of five tasks with a set of compute resources (PC1, PC2, PC3) interconnected with varying bandwidth and having its own storage unit (S1, S2, S3) (Figure 4.11(a)) and its sample particle (Figure 4.11(b)). The heuristic proposed is generic as it can be used for any number of tasks and resources by simply increasing the dimension of the particles and the number of resources, respectively.

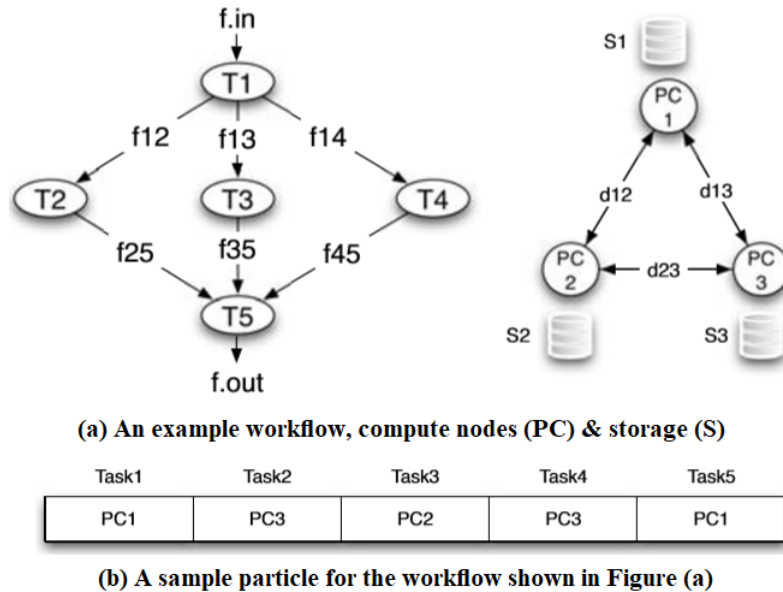


Figure 4.11: Scheduling of a workflow with required resources [15]

Su et al. [54] presented a cost-efficient task-scheduling algorithm using two heuristic strategies. The first strategy dynamically maps tasks to the most cost-efficient VMs based on the concept of Pareto dominance. The second strategy, a complement to the first strategy, reduces the monetary costs of non-critical tasks. Zeng et al. [55] addressed the problem of scheduling Many Tasks Workflow (MTW) application on

clouds. They presented a budget-conscious scheduling algorithms, which is cared by both cloud service providers and users. Towards this end, they proposed a budget conscious scheduler that is shown to improve the overall makespan and resource utilization of MTW applications in cloud.

Several algorithms have been proposed for workflow scheduling, but most of them fail to incorporate the key features of cloud like heterogeneous resources, pay-per-usage model, and elasticity along with the QoS requirements. Kaur et al. [57] proposed a hybrid genetic algorithm which uses the Predict Earliest Finish Time (PEFT) to generate a schedule as a seed with the aim to minimize cost while keeping execution time below the given deadline. A good seed helps to accelerate the process of obtaining an optimal solution. The algorithm is simulated on WorkflowSim and is evaluated using various scientific realistic workflows of different sizes.

Visheratin et al. [56] dealt with scheduling scientific workflows in heterogeneous cloud-based computational environment. They considered the main features of IaaS providers, like wide variety of offered computational services and pay-as-you-go price model with time periods of charge. More specifically, Visheratin et al. [56] proposed an heuristic algorithm, named Levelwise Deadline Distributed Linewise Scheduling (LDD-LS), for scheduling workflows in hard-deadline constrained clouds. After, they combined LDD-LS with the implementation of IC-PCP algorithm to initialize their proposed metaheuristic algorithm called Cloud Deadline Coevolutional Genetic Algorithm (CDCGA).

Zhou et al. [58] tackled the problem of determining the tasks execution order, task-to-VM allocation, and VM type assignment. The goal of the work is to design algorithms that optimize execution cost and makespan for workflows scheduled in hybrid clouds. To this end, they formulated 2 workflows scheduling problems. The first is a single-objective optimization problem that aims to minimize execution cost under deadline constraint. The second is a multi-objective optimization problem that attempts to minimize execution cost and makespan simultaneously. To solve these two problems, authors designed two GA-based heuristic algorithms. Precisely, they proposed a single-objective workflow scheduling optimization approach called DCOH (Deadline-constrained Cost Optimization for Hybrid clouds) for minimizing the monetary cost of scheduling workflows under deadline constraint. Based on DCOH, they further proposed a multi-objective workflow scheduling optimization approach, called MOH (Multi-objective Optimization for Hybrid clouds), to optimize makespan and cost of scheduling workflows simultaneously.

As mentioned in [53], nowadays scientists and companies are confronted with multiple competing goals such as makespan in high-performance computing and economic cost in clouds that have to be simultaneously optimised. In contrast to the most existing approaches that aggregate all objectives in a single function, Durillo et al. [53] proposed a new method called MultiObjective Heterogeneous Earliest Finish Time (MOHEFT) as an extension to the well-known HEFT [264] monoobjective workflow scheduling algorithm. Thus, authors offered a method for optimizing the makespan

and economic cost of running workflow applications in an Amazon-based commercial cloud. Tan et al. [52] developed specific scheduling strategies to effectively optimize time, cost, and trust factors in enterprise systems. More precisely, they designed cloud service selection algorithms to form optimum workflow application while meeting different constraints from users. In order to address the problem of workflows in an optimal and reliable way, a timed workflow model is proposed to meet the requirements of enterprise system integration.

Even so, we observe a clear missing of the consideration of the variety of pricing strategies of cloud resources to propose a scheduling plan, that provides the optimal deployment cost of a time-aware BP. Though, in our work, we optimize the BP deployment cost while taking into account different pricing strategies (on-demand, reserved, spot), various cloud providers, and the BP constraints (time, capacity).

4.3.3 Pricing strategies

Several works have been done for pricing strategies in the IT service market [265–270]. For example, Gajananan et al. [266] extended their work [265] by proposing a top-down pricing approach of IT service deals. More specifically, in [265], based on both market and historical data, the proposed algorithm estimates the prices of the highest level of services included in a solution. Nevertheless, in [266], authors suggested a new pricing approach that determines the price points of new complex deals using the secondary service level. The new approach automatically generates the expected input at lower level from the input for the highest level of the services of the deals to be priced. Consistent with the findings in [265], the experimental results, done by Gajananan et al. [266], have shown that using historical data is more accurate than using market data to estimate the prices for many services.

Basu et al. [267] developed optimal pricing strategies for a typical cloud service provider. To this end, they modeled the utility of a customer of cloud services as a function of a set of parameters which positively and negatively effect the utility of a end user. Moreover, they explored two pricing plans: usage based and fixed fee plan and determined the conditions under which end users would select one plan over another. Finally, they discussed the significance of these conditions for cloud service providers.

Ibrahim et al. [268] studied the pricing fairness on the pay-as-you-go charging, and then introduced the pay-as-you-consume strategy to resolve the unfairness in the current pay-as-you-go pricing scheme. While the pay-as-you-consume strategy seemingly reduces the cloud providers' profit, it urges providers to improve their system design and optimization to provide good services and to gain competitive advantages. They have demonstrated that the predication of the proposed strategy can achieve up to 90% accuracy regardless the VM consolidations or the I/O workloads.

Arshad et al. [269] proposed a literature review of different pricing scheme with respect to their advantages, limitations, and possible future directions, especially for

vendors to seek new research directions in dynamic pricing schemes. Besides, Samimi et al. [270] provided a comparative review of grid and cloud computing economic and pricing strategies from which appropriate tariffs and charging strategies can be chosen to meet particular business objectives. More precisely, authors gave the basic core principles and a comparative review of the latest and most appropriate economic and pricing strategies applicable to grid and cloud computing in order to propose better models for the future.

However, in the analyzed papers, we did not find any work that provides a tool that allows the users to analyze dynamically the resources and services costs. Only recently, Alves et al. [271] have proposed an extension of CloudSim (called CMCloudSim) features to model and simulate the cost of cloud resources from Amazon, Google, and Microsoft Azure. The proposed extension retrieves the cost from the website of these providers. This extension does not support the different pricing strategies of Amazon (i.e., on-demand, reserved instances, spot blocks, and spot instances). It focuses only on the on-demand strategy.

Also, various research studies have been addressing the topic related to the deployment of BPs in cloud environments. In [272, 273], the authors discussed the benefits and drawbacks of blending BPM and cloud computing. To the best of our knowledge, except the extension proposed in [274], there is no work done around the simulation of cloud resources allocated for a BP. In [274], the authors proposed an extension of CloudSim to simulate the performance in terms of time and cost of the allocated cloud resources. Nevertheless, the extension does not support the temporal constraints and simulation of cloud tasks based on the BP control-flow.

Unlike the researches detailed above, in this thesis, we propose an extension of the famous cloud simulator provided in the market, CloudSim [69], to assist the BP designer to simulate a cloud resource allocation and to have an estimation of its real cost. This extension is related to the capability to simulate the cloud resources consumed in the BP model.

4.3.4 Synthesis

Table 4.3 summarizes some existing approaches related to resource allocation optimization in the literature based on the criteria relevant to our work. Many of these approaches take into account the resource perspective in BP, wherein some of them consider only the human resource perspective in the BPM domain. We cluster the above-mentioned approaches based on the criteria important to our thesis: (1) control-flow perspective, (2) resource, (3) assignment, (4) scheduling, and (5) technique. In Table 3.3, we use the following nomenclature: "+" to depict that the approaches satisfy the corresponding criteria, "-" to depict that they do not satisfy the corresponding criteria, "+/-" to depict that the approaches partially satisfy the corresponding criteria.

As visible from Table 4.3, the majority of the approaches presented focus on

Table 4.3: Summary of the literature study of resource allocation optimization

Work	Control-flow perspective	Resource	Assignment	Scheduling	Technique
[43]	+	human	+	-	Algorithm
[44]	+	human	+	-	SAHMM
[228, 233]	-	human	+	-	Algorithms
[14]	+	human	-	+	ASP
[234]	-	human	-	+	MH Algorithm
[135, 136, 235]	-	human	-	+	Integer program
[45, 46]	+	cloud	+	-	Linear program
[47–49, 51, 262]	+	cloud	+	-	Algorithms
[229, 255, 256, 259]	-	cloud	+	-	Algorithms
[253]	-	cloud	+	-	Integer linear program
[254]	-	cloud	-	-	Stochastic program
[52–55]	+	cloud	-	+	Algorithms
[15, 56]	+	cloud	-	+	Heuristic algorithms
[57, 58]	+	cloud	-	+	Genetic algorithms
Our approach	+	cloud	+	+	Linear program

optimizing resource allocation in BPM context [14, 15, 43–49, 51–58]. A lot of interest is given for human resources while they are crucial in BPM as they are responsible for process execution or supervision [206]. Thus, there is a need for such approaches as domain-specific processes (e.g. health care) involving humans are scarce and costly. However, our work is mainly focused on optimizing cloud resources into BPs.

Other approaches focus on cloud resources in BPM domain. This is because cloud computing becomes an interesting infrastructure for enterprises that are migrating all or some of their process activities into cloud. Besides, we note that authors propose to find an optimal resource assignment [43–49, 51, 229, 233, 253, 255, 256, 258, 259, 262] or an optimal scheduling [14, 15, 52–58, 135, 136, 234, 235] using various techniques such as algorithms (genetic/ heuristic), programming models (linear/integer). Those approaches are categorized into four different viewpoints as seen in Table 4.5: (1) an approach considering resource temporal availabilities [57], (2) approaches considering pricing strategies to reduce process deployment costs [48, 54, 56, 58], (3) approaches considering activities temporal constraints [53]. Overall, the work on optimizing cloud resources in BPM context propose assignment or scheduling methods.

Each optimization problem has a specific set of constraints such as budget constraint, makespan constraint, temporal constraints, resource temporal availabilities,

Table 4.4: Summary of the literature study of human allocation optimization

Work	Control-flow perspective	Activities temporal constraints	Resource temporal availabilities
[13, 228]	+	+/-	+
[14, 43]	+	+/-	-
[50]	+	-	-
[135, 136, 233, 235]	-	+/-	+
[234]	-	-	+

etc. So, in Table 4.4 we cluster the approaches that study optimization of human resource allocation and in Table 4.5 we cluster the approaches that study optimization of cloud resource allocation. In Table 4.4, the approaches are classified based on three criteria: (1) control-flow perspective, (2) activities temporal constraints, and (3) resource temporal availabilities. Whereas, in Table 4.5 the approaches are classified based on four criteria: (1) control-flow perspective, (2) activities temporal constraints, (3) resource temporal availabilities, (4) pricing strategies, and (5) Simulation tool. As visible in Table 4.3, some of the authors work on process perspective. Further, most of the authors take into consideration human resource temporal availabilities. The majority of these authors suggest that activities have some temporal constraints such as duration. But, in Table 4.5, authors working on cloud computing concentrate on BPM domain where BP are timed-aware. But, cloud resource temporal availabilities is taken into account by few authors. Overall, the works involve pricing strategies but the majority consider only pay-as-you-go strategy. Besides, cloud resources are considered always available and do not have temporal constraints. Further, few authors use simulation tools such as WorkflowSim and CloudSim to simulate the cloud resource allocation in order to validate the proposed approaches. Authors do not specify advanced temporal constraints for activities such as temporal dependency. Contrary to above, our work is focused on optimizing cloud resource allocation cost under constraints such as control-flow perspective, activities temporal constraints, resource temporal availabilities and pricing strategies constraints. Moreover, compared to the proposed extension for cloud simulators presented in Section 4.3.3, in our work, we extend CloudSim to support the temporal constraints, pricing strategies, and simulation of cloud tasks based on the BP control-flow.

4.4 Conclusion

In this chapter, we presented the current state-of-the-art with respect to our research problems detailed in Chapter 1. We reviewed approaches in the literature for developing cloud-aware BPs, i.e., specification, verification and optimization of cloud resource allocation in BP models. To this end, we divide the related work based on our re-

Table 4.5: Summary of the literature study of cloud resource allocation optimization

Work	Control-flow perspective	Activities temporal constraints	Resource temporal availabilities	Pricing strategies	Simulation tool
[57]	+	+/-	+	+	+ WorkflowSim
[48, 54]	+	+/-	-	+	+ CloudSim
[56, 58]	+	+/-	-	+	-
[53]	+	+/-	-	-	-
[46, 47, 52]	+	+/-	-	-	-
[45, 49, 262]	+	-	-	-	-
Our approach	+	+	+	+	+ CloudSim

search questions into two main groups: (1) supporting the cloud resources' temporal constraints in BP, and (2) optimization of BP deployment cost in cloud resources. In the first group, we reviewed the approaches available in the literature from the following perspective: (1) formal specification in BP models, and (2) formal verification in BP models. In the second group, we studied and reviewed the approaches from the following perspective: (1) human resource allocation, and (2) cloud resources allocation. We evaluate and compare the existing approaches to understand the missing parts of the approaches in the state-of-the-art, which are (1) lack of support of cloud in time-aware BPs, especially temporal constraints, (2) lack of optimizing time-aware BP deployed in cloud resources offered in various pricing strategies.

To bridge the above-mentioned research gap, we describe our contribution in the following chapters. In Chapter 5, we detail our first contribution towards supporting cloud resources' temporal constraints in BPs that enables the formalization, modeling, and verification of time-aware cloud resource allocation in BPs. In Chapter 6, we tackle the issues related to the optimization of time-aware cloud resource allocation in BPs, as the current state-of-the-art does not support it.

Supporting Cloud Resources Temporal Constraints in BPs

Contents

5.1	Introduction	95
5.2	Graphical Modeling	96
5.2.1	Formal definitions	96
5.2.2	BPMN extension	99
5.3	Model transformation	102
5.3.1	Transformation: from BPMN to timed automata	102
5.3.2	Automatic transformation	107
5.3.3	UPPAAL Meta-Model	109
5.4	Correctness analysis	109
5.5	Evaluation	111
5.5.1	Supporting pricing strategies description	111
5.5.2	BPMN model transformation	112
5.5.3	Checking CTL properties	115
5.6	Conclusion	118

5.1 Introduction

In this chapter, we present our approach for correct cloud resource allocation in time-aware BPs. As mentioned in Chapter 2, pricing strategies are defined based on temporal constraints. Moreover, the organisations' BPs are time constrained. Further, to be performed, activities may consume cloud resources proposed under various pricing strategies. But, to correctly utilize them the temporal constraints of cloud resources should match the temporal constraints of BP activities. For instance, if the resource temporal availability [62] does not cover the activity temporal duration this may lead to a deadlock. Therefore, organizations need to design their BP models enriched with

both control flow and resource temporal constraints specifications using a formal language. Thus, they can formally check the matching between temporal constraints of both: activities and cloud resources to ensure that BPs can be correctly performed.

To address the above challenges, our goal is to define a formal model of a BP enriched with temporal constraints, cloud resources, and pricing strategies. The aim of this formal specification is to avoid temporal conflicts. In fact, formal languages have shown their usefulness in the modeling of correct systems (e.g. timed automata [106], Petri nets [107], process algebra [108], Event-B [109], etc). The specification and reasoning about system properties is ensured by the mathematical foundations and formal logic. But, BP designers may do not have a strong background in formal languages and do not master well verification tools. Therefore, we rely on the broadly accepted BP modeling language, BPMN, and we take it as our stating modeling language and we extend it to describe cloud resources and their pricing strategies. While BPMN is a semi-formal language we transform the extended BP models, designed in BPMN, into a network of timed automata in order to verify the temporal correctness of resource allocation.

Concretely, first, we propose a model of cloud resource allocation in time-aware BP. Second, taking the advantage of its extensibility, we propose an extension to BPMN to integrate the cloud resource allocation in time-aware BP models. Second, we propose a set of transformations rules that we implemented based on MDE, to generate automatically from BPMN extended models a network of timed automata [106]. The latter will be the inputs to the model checker UPPAAL to formally verify the non-conflict between temporal constraints of both activities and cloud resources. Our approach is implemented as an Eclipse plug-in and evaluated using a real case study from France Telecom Orange labs.

The contributions presented in this Chapter were published in [62–64]. The remainder of this Chapter is organized as follows: We start by presenting our graphical modeling step in Section 5.2.1. Then, we explain our verification step in Section 5.4. Next, we depict the evaluation of our approach in Section 5.5. Finally, we present our conclusion in Section 5.6.

5.2 Graphical Modeling

We detail in this section our first contribution. Namely, we start by presenting our proposed formal definitions. Then, we move to detail our BPMN extension.

5.2.1 Formal definitions

In this section, we present our proposal to formally define cloud resources, their pricing strategies, as well as activities' temporal constraints. We start by presenting a formalization of cloud resources (Section 5.2.1.1) and BP model (Section 5.2.1.2). Then, we explain our BPMN extension (Section 5.2.2).

5.2.1.1 Cloud resources in BP

Cloud computing is known for *aaS model with focus here on computing resources exposed as part of the Infrastructure-as-a-Service (IaaS) model for not increasing the complexity of the cloud resource management. A computing resource has a unique identifier and predefined capacities. We formally define a cloud resource as follows (see Definition 5.2.1):

Definition 5.2.1. *A cloud resource is a couple (id, Cap) where:*

- *id is a unique identifier;*
- *Cap defines a resource' capacities in terms of memory amount and virtual core number, $Cap=(RAM, vCPU) \in \mathbb{N} \times \mathbb{N}$.*

Various efforts are made for the enhancement of cloud resource description in BPM [33,139,140]. However, the cloud resource description remains poorly operated due to the lack of a formal definition of pricing strategies that are specified in natural languages. So, in the following, we present a formal definition of a cloud resource pricing strategy (see Definition 5.2.2).

Definition 5.2.2. *A cloud resource pricing strategy is a triple $st=(type,TC,c)$ where:*

- *type is a type of strategy;*
- *TC is the temporal availability of a price that the strategy st imposes;*
- *c is a unit hourly cost that the strategy st proposes, $c \in \mathbb{R}$.*

As described in Chapter 2, a cloud resource has a limited temporal availability which restricts the time span allowed for using a cloud resource R at a defined price. Temporal availabilities over a cloud resource's pricing strategy are of 2 types:

- Relative temporal constraints specify a time interval $[MinAvR, MaxAvR]$ in which a resource is available at a certain price; $1 \leq MinAvR \leq MaxAvR$.
- Absolute temporal constraints specify the start and finish times of a resource availability at a certain price. Those constraints could be specialized into: Start Using No Earlier Than ($SUNET(R)$), Finish Using No Earlier Than ($FUNET(R)$), Start Using No Later Than ($SUNLT(R)$), and Finish Using No Later Than ($FUNLT(R)$).

As mentioned in Section 2.3.2, we consider 4 pricing strategies: on-demand, reserved, spot instance and spot block. We present in Table 5.1 which temporal specification can be used in each pricing strategy. While, reserved and spot block are continuously available for a specified temporal duration, so they can be described with $MinAvR$ and $MaxAvR$ (i.e., relative). Otherwise, to specify a spot instance, that has an interruption risk, $SUNET$, and $FUNET$ (i.e., absolute) can be used.

Table 5.1: Temporal availabilities of cloud resources

	Relative		Absolute	
	MinAvR	MaxAvR	SUNET	FUNET
On-demand				
Reserved	+	+		
Spot instance			+	+
Spot block	+	+		

However, on-demand instances are always available, so we do not need to specify a temporal constraint for cloud resources.

Table 3.2 (resp., Table 3.3) presents a set of Amazon (resp., Microsoft) cloud resources ($\mathcal{R}=\{R_1, R_2, R_3, R_4, R_5\}$). Their operating system is Linux and availability time zone is us-east-a1. In our current work, we assume that the size of transferred data between cloud resources is not considered and is left as future work.

Amazon EC2 offers different instance types with different computing capacities ($vCPU$) and (RAM) and grouped into instance families. For instance, $R_3 = m3.2xlarge$ corresponds to an Amazon computing resource of type $m3$. The latter belongs to *general purpose* instance family that provides a balance of compute and memory resources, and can be used for a variety of workloads [4]. Still in Table 3.2, the unit hourly price of R_3 as spot instance is equal to $c_{313}=0.372\$/h$. As a result, R_3 is temporally available for a minimum duration $MinAvR_3$ of 1 hour and a maximum duration $MaxAvR_3$ of 6 hours (i.e., $st_{313}=(spot, [1h,6h], 0.372\$/h)$).

5.2.1.2 Business process model

A BP consists of a set of activities A (where $A = \{a_q : q \in \{1, \dots, z\}\}$) that are performed in coordination in an organizational and technical environment to jointly achieve business goals [275]. It happens that a BP is subject to some temporal constraints which can be relative or absolute. For more details about temporal constraints, we refer readers to [31]. Activities are also subject to financial penalties price [137,138] when they are canceled due to resource interruption.

Definition 5.2.3. *Business process model is a tuple (N, E, F, Req_A, Pen_A) where:*

- N is the set of nodes that correspond to activities A , gateways G , and events Ev , i.e., $N=\{A \cup G \cup Ev\}$;
- $E \subseteq N \times N$ is the set of edges;
- $F : A \longrightarrow T$ is a function that assigns temporal constraints T to activities A ;
- Req_A is the set of activities requesting cloud resources;

- Pen_A is the set of financial penalties that are subject to when activities are canceled due to the interruption of a resource leased as a spot instance.

For example, Figure 3.3 presents the service supervision BP of France Telecom/O-range. Table 3.1 shows its activities temporal constraints, capacity requirements, and financial penalties prices.

Definition 5.2.4. *An activity a is a tuple $(temp, res, type)$ where:*

- $temp$ is a set of temporal constraints which can be temporal duration, dependency, and/or absolute temporal constraints;
- res is the resource allocated;
- $type$ is the pricing strategy type.

For instance, $a_1 = (\{Duration((a_1, 1, 2))\}, R_1, \text{on-demand})$ in the “supervision process” BP means that a_1 has a temporal constraint expressed in term of minimum and maximum duration. Besides, it consumes R_1 as an on-demand instance.

In [276], authors proposed a survey of time-related aspects of process models. They identified the existing approaches that specify and verify temporal constraints and enhance the time dimension in the BPM field. Some approaches opted for modeling time using graphs while others used formal specification languages and algebras (e.g., Linear Temporal Logic [277], Allen’s algebra [29], and time petri nets [278]) for specifying and verifying time-based BP models. Based on this research work, we compare our time representation to Allen’s algebra. When using Allen’s algebra, we identified that the BP designer can not specify that a BP activity should be executed with respect to a specific time nor a date. So, compared to Allen’s algebra, we can represent and specify absolute temporal constraints including MSO/MFO (Must Start/Finish On a defined date), SNLT/FNLT (Start/Finish No Earlier Than). These constraints mean that a BP activity should be executed in a defined date.

5.2.2 BPMN extension

We move now to the BPMN extension step. The BPMN notation is extensively used in the BPM community for BP modeling [80]. BPMN 2.0 is its latest version. It has several concepts including resource perspective. However, it does not support cloud resource temporal availabilities related to pricing strategies and advanced temporal constraints for BP activities.

Based on our model (Section 5.2.1), we propose two main extensions (Figure 5.1 and Figure 5.8). The first proposed extension (Figure 5.1) is with cloud resources (particularly VMs) and their pricing strategies. The second proposed extension (Figure 5.8) is with temporal constraints related to BP activities. We notify that the classes in grey color are the BPMN 2.0 predefined meta-classes. The classes in white

color are new meta-classes extending BPMN. We propose for cloud resource extension, an extension meta-class “ResourceExtension” to the BPMN meta-class “Resource” [21]. A resource extension can be a VM (VirtualMachine). A VM can be related to four main types of pricing strategies which are on-demand, reserved, spot block, and spot instance. As we mentioned in Chapter 2, each pricing strategy has a cloud resource’s temporal availability [62]. The reserved pricing strategy and the spot block require relative temporal constraints. It is expressed by *MinAvR* and *MaxAvR* (refer respectively to the Minimum and Maximum Resource temporal Availability) (Section 5.2.1.1). Moreover, the reserved pricing strategy is characterized by paying in advance for an instance, reserving it for 1 or 3-year period. The payment can be with different options [4]: All-upfront, partial-upfront, or no-upfront depicted by *PaymentOption* as an enumeration class. Otherwise, the spot instance has absolute temporal constraints which are SUNET and FUNET (Section 5.2.1.1). However, for on-demand, we do not need to specify temporal constraints for cloud resources.

Based on [29–31, 73], we propose activities’ temporal constraints extension such as Duration, MSO, MFO, SNLT and FNLT (respectively Must Start/Finish On, Start/Finish No Later Than) (Section 2.2). To define the latter constraints, we propose a meta-class extension called “ExtensionEventDefinition” to the existing BPMN meta-class “EventDefinition”. An extension meta-class called “ExtensionFlowElement” extends “FlowElement” BPMN metaclass to define the temporal dependency constraint. In addition, we extend the BPMN Process meta-class with an extension meta-class called “ProcessExtension” in order to define “ProcessStartTime” meta-class. Above all, we propose a meta-class extension “ExtensionCloudTask” to exhibit our extension to the BPMN meta-class “Task”. This extension shows that the Task flow requires a cloud resource with a chosen pricing strategy.

Listing 5.1: An excerpt of the xsd document to extend BPMN

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
3   <xs:element name="ResourceTimeExtension">
4     <xs:complexType>
5       <xs:sequence>
6         <xs:element name="Pricing.Strategy" type="Pricing.Strategy"/>
7         <xs:element name="Resource">
8           ...
9         </xs:element>
10        <xs:element name="ResourceExtension" type="ResourceExtension"/>
11        <xs:element name="VirtualMachine">
12          <xs:complexType>
13            <xs:complexContent>
14              <xs:extension base="ResourceExtension">
15                ...
16                <xs:attribute name="price" type="xs:float">
17                  </xs:attribute>
18                </xs:extension>
19              </xs:complexContent>
20            </xs:complexType>
21          </xs:element>
22          ...

```

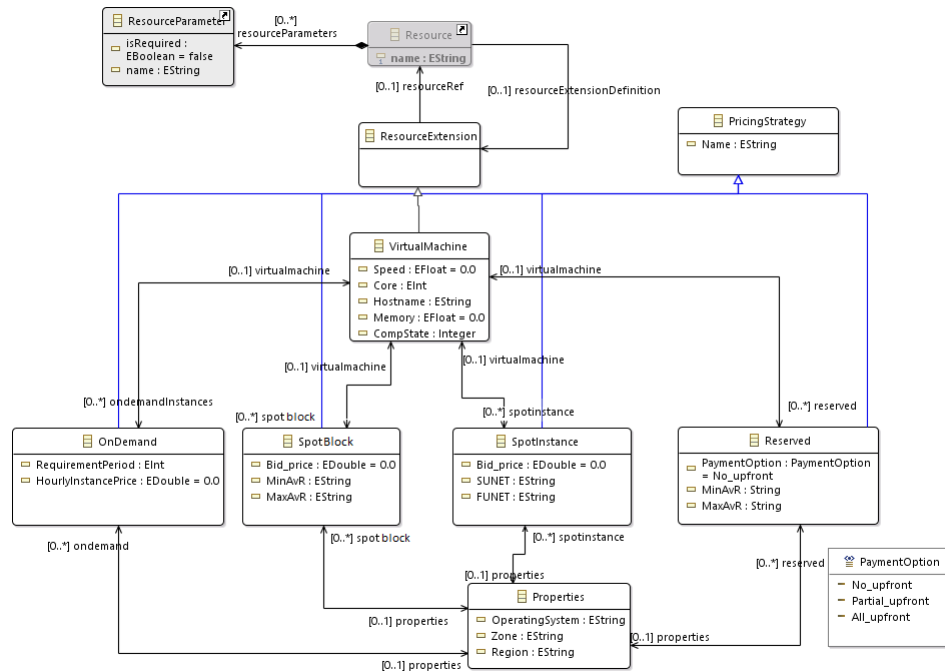


Figure 5.1: The extension of resource element in BPMN

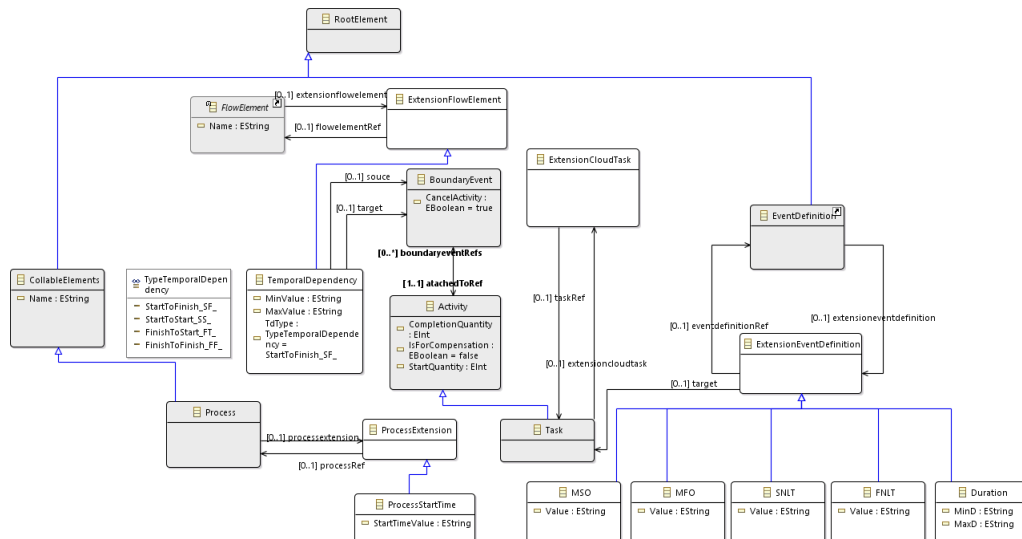


Figure 5.2: BPMN temporal extension

```

23     <xs:element name="SpotBlock">
24         <xs:complexType>
25             <xs:complexContent>
26                 <xs:extension base="Pricing_Strategy">
27                     <xs:attribute name="hourlyspotprice" type="xs:double">
28                         </xs:attribute>
29                     <xs:attribute name="MinAvR" type="xs:string">
30                         </xs:attribute>
31                     <xs:attribute name="MaxAvR" type="xs:string">
32                         </xs:attribute>
33                 </xs:extension>
34             </xs:complexContent>
35         </xs:complexType>
36     </xs:element>
37     ...
38 </xs:sequence>
39 </xs:complexType>
40 </xs:element>
41 <xs:complexType name="Pricing_Strategy">
42     <xs:attribute name="Name" type="xs:string">
43     </xs:attribute>
44 </xs:complexType>
45 <xs:complexType name="ResourceExtension">
46 </xs:complexType>
47 ...
48 </xs:schema>

```

The class diagram shown in Figure 5.1 is translated to an xsd document as shown in Listing 5.1. The “VirtualMachine” element (Line 11) has a complex type “ResourceExtension”. So, it will have all the attributes and structure of “ResourceExtension” but with additional attributes such as price. Besides, the “SpotBlock” has a complex type that extends the type “PricingStrategy”, and adds to it attributes including “hourlyspotprice”, “MinAvR”, “MaxAvR”. This xsd extension is imported in the xsd of BPMN 2.0.

5.3 Model transformation

In this section, we detail our transformation step. We start by presenting how we transform BPMN to timed automata model. Then, we present how we automate our transformation step.

5.3.1 Transformation: from BPMN to timed automata

Turning now to the transformation step. To this end, we present in this section a set of rules to automatically transform time-aware BPMN models into a network of timed automata. To this end, we present namely our transformation rules, then the automatic transformation step, and finally a description of the UPPAAL meta-model.

We recall that our BPMN process model consists of activities that are subject to temporal constraints and consume cloud resources. We developed a set of transformation rules that take as an inputs a BP’s activities along with the cloud resources

they consume and produce as an output a network of timed automata depending on these resources' pricing strategies.

A network of timed automata consists of: (i) the BP's timed automata with focus on its activities' states and temporal constraints and (ii) the cloud resources' set of timed automata that these activities will consume. The synchronisation of BP-related and resource-related timed automata is achieved using binary channels.

To begin with, let's consider an activity a that consumes a cloud resource r . Because of this resource's pricing strategies, we identify four consumption cases: on-demand instance, reserved or spot blocks, spot instance, and shareable instance. The last case is used for a spot block instance that has a fixed shareable capacity. We also consider that although our BPs could be subject to temporal dependency and absolute temporal constraints, we only look into duration constraints.

For more details about how other constraints are handled, readers are referred to [30].

- **On-demand instance:** When a is defined as $(\{\text{Duration}(a, \text{Min}D_a, \text{Max}D_a)\}, R, \text{on-demand})$, a is transformed into two timed automata, TA_a and TA_{rd} . On the one hand, TA_a , represents the activity's timed automata and consists of three locations: a_{Ready} , $a_{Working}$, and a_{Finish} . a_{Ready} means that a is ready for execution, $a_{Working}$ means that a is running and consumes R as an on-demand instance, and a_{Finish} means that a has executed successfully allowing to free R . The transition from a_{Ready} to $a_{Working}$ initializes a clock t to zero. And, the transition from $a_{Working}$ to a_{Finish} takes a guard to control the activity's temporal duration.

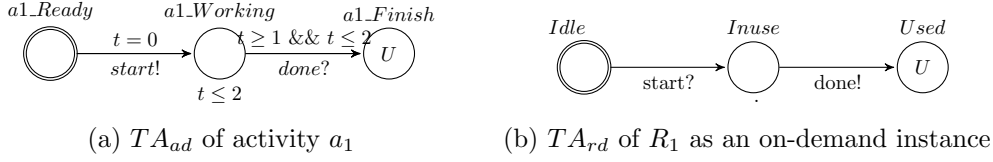
Formally, see Definition 2.5.1, TA_a is a tuple $L=\{a_{Ready}, a_{Working}, a_{Finish}\}$, $l_0 = a_{Ready}$, $X = t$, $I(a_{Ready})=\emptyset$, $I(a_{Working})=\{t \leq \text{Max}D_a\}$, $I(a_{Finish})=\emptyset$, and $\text{Tr}=\{(a_{Ready}, \text{start!}, t = 0, \emptyset, a_{Working}), (a_{Working}, \text{done?}, t \geq \text{Min}D_a \ \&\& \ t \leq \text{Max}D_a, a_{Finish})\}$.

On the other hand, TA_{rd} consists of three locations: $Idle$, $Inuse$, and $Used$. $Idle$ means that R is available but not-assigned, yet; $Inuse$ means that R is assigned to a and is in-use; and, $Used$ means that R was consumed in the past by a and now is free.

Formally, see Definition 2.5.1, TA_{rd} is a tuple $L=\{Idle, Inuse, Used\}$, $X = \emptyset$, $l_0=Idle$, $I(Idle)=\emptyset$, $I(Inuse)=\emptyset$, $I(Used)=\emptyset$, and $\text{Tr}=\{(Idle, \text{start?}, \emptyset, Inuse), (Inuse, \text{done!}, \emptyset, Used)\}$.

For illustration purposes, we consider $a_1=(\{\text{Duration}((a_1, 1, 2)\}, R_1, \text{on-demand})$ in the "supervision process" BP. a_1 is transformed into two timed automata: TA_a is shown in Figure 5.3a and TA_{rd} is shown in Figure 5.3b. a_1 can start consuming R_1 only if the clock t is less than 2 (i.e., defined as an invariant) and will execute successfully only if its temporal duration is met (i.e., defined as a guard).

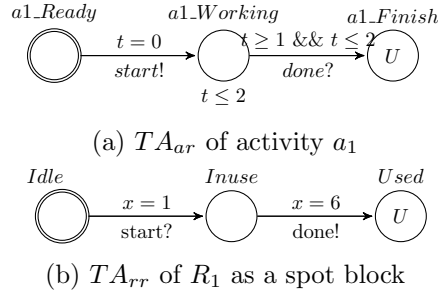
- **Reserved or Spot blocks:** When a is defined as $(\{\text{Duration}(a, \text{Min}D_a,$

Figure 5.3: Allocation of R_1 as an on-demand instance cloud-resource to activity a_1

$MaxD_a$ }), R , reserved) or ($\{Duration(a, MinD_a, MaxD_a)\}$, R , spot blocks) it is transformed into two timed automata, TA_{ar} and TA_{rr} . TA_{ar} and TA_{ad} are similar. Further, TA_{rd} and TA_{rr} have the same formalism in terms of locations, transitions, clock, guard, and invariant. This is due to the fact that a cloud resource consumed as on-demand, reserved or spot blocks, is not subject to any interruption once it becomes consumed. Thus, the resource's timed automata, TA_{rr} , consists of the same three locations as TA_{rd} : $Idle$, $Inuse$, and $Used$. But, they are different in terms of transitions since R can have temporal duration. As a result, we assign a clock x to the transitions between $\{Idle, Inuse\}$ and $\{Inuse, Used\}$ to specify the minimum ($MinAvR$) and maximum ($MaxAvR$) durations of R 's temporal availability.

Formally, see Definition 2.5.1, TA_{rr} is a tuple where $L = \{Idle, Inuse, Used\}$, $X = \{x\}$, $l_0 = Idle$, $I(Idle) = \emptyset$, $I(Inuse) = \emptyset$, $I(Used) = \emptyset$, and $Tr = \{(Idle, start?, \{x = MinAvR\}, Inuse), (Inuse, done!, \{x = MaxAvR\}, Used)\}$.

For illustration, we refer to our case study where Figure 5.4a, same as Figure 5.3a, presents a_1 's timed automata TA_{ar} , and Figure 5.4b presents R_1 's timed automata TA_{rr} . When R_1 takes on $Inuse$ state, the clock x is initialized to 1. If it is initialized to 6 that is the maximum duration, then the resource is considered as used and no longer available.

Figure 5.4: Resource allocation with R_1 as spot block to an activity a_1

- **Spot instance** When a is defined as ($\{Duration(a, MinD_a, MaxD_a)\}$, R , spot instance), a is transformed into two timed automata TA_{aa} and TA_{ra} . It is worth

noting that R here can be interrupted while it is under use. On the one hand, TA_{aa} is the activity's timed automata. So, activity a would be blocked. That is why we add a fourth state, $a_{Blocked}$, reached when the resource R becomes interrupted. Defining a Boolean variable e is required to verify if an external event has happened (i.e., another customer proposes a higher bid price and takes resource R). Thereby, transiting from $a_{Working}$ to $a_{Blocked}$ is enabled only if the guard indicating the interruption is true.

Formally, see Definition 2.5.1, TA_{aa} is a tuple where $L=\{a_{Ready}, a_{Working}, a_{Blocked}, a_{Finish}\}$, $l_0=a_{Ready}$, $X = \{x\}$, $I(a_{Ready})=\emptyset$, $I(a_{Working})=\{x \leq MaxD_a\}$, $I(a_{Blocked})=\emptyset$, $I(a_{Finish})=\emptyset$, $E=\{(a_{Ready}, start!, x = 0, \emptyset, a_{Working}), (a_{Working}, done?, e == true, a_{Blocked}), (a_{Working}, done?, x \geq MinD_a \ \&\& \ x \leq MaxD_a, a_{Finish})\}$.

On the other hand, TA_{ra} is composed of four states: *Idle*, *Inuse*, *Used*, and *Interrupted*. If the spot price becomes greater than the bid price, R will be interrupted. In this case, the transition from *Inuse* to *Interrupted* is enabled. So we use a Boolean variable e to control the interruption. Moreover, to satisfy the absolute constraint over R , we initialize x to zero and take it as a timed reference to subsequently specify that exactly $FUNET - SUNET$ hours must separate the starting and finishing availability times of the resource. Therefore, to transition from *Inuse* to *Used*, we update x based on the difference between $FUNET$ and $SUNET$.

Formally, see Definition 2.5.1, TA_{ra} is a tuple where $L=\{Idle, Inuse, Interrupted, Used\}$, $l_0=Idle$, $X = \{x\}$, $I(Idle) = \emptyset$, $I(Inuse)=\emptyset$, $I(Interrupted)=\emptyset$, $I(Used)=\emptyset$, and $Tr=\{(Idle, start?, \{x=0\}, Inuse), (Inuse, done!, \{x=FUNET-SUNET\}, Used), (Inuse, e == true, \emptyset, Interrupted)\}$.

For illustration purposes, we consider that $a_1=(\{Duration((a_1, 1, 2)\}, R_1, \text{spot instance})$ in the ‘‘supervision process’’ BP. Figure 5.6 presents the transformation output of a_1 . Figure 5.5a is different from Figure 5.4b; $a_{1_{Blocked}}$ location is reached when R_1 is interrupted (i.e., defined as a guard). The timed automata of R_1 is illustrated in Figure 5.5b. When R_1 is in state *Inuse* the clock x , taken as a time reference, is initialized to zero. Then, it is initialized to 10 if its finish availability time is reached. The Boolean variable e is ‘‘true’’ to indicate that resource R_1 is interrupted (i.e., defined as a guard).

- **Shareable instance:** Let consider a set of activities $A_{sh}=\{a_m, \text{ where } m \in \{1 \dots z\}\}$ defined as $a_m (\{Duration(a_m, MinD_{a_m}, MaxD_{a_m})\}, R, \text{reserved})$ or $(\{Duration(a_m, MinD_{a_m}, MaxD_{a_m})\}, R, \text{spot blocks})$ that consume the same resource R as reserved or spot blocks instance but not at the same time. They are transformed into $z + 1$ timed-automata, $TA_{ash_1}, TA_{ash_2}, \dots, TA_{ash_m} \dots, TA_{ash_z}$ and TA_{rsh} . On the one hand, TA_{ash_m} (where $m \in \{1 \dots z\}$) present the timed automata of each activity a_m . All the TA_{ash_m} have the same formalisms in terms of locations, transitions, clocks, guards and invariants. Every TA_{ash_m}

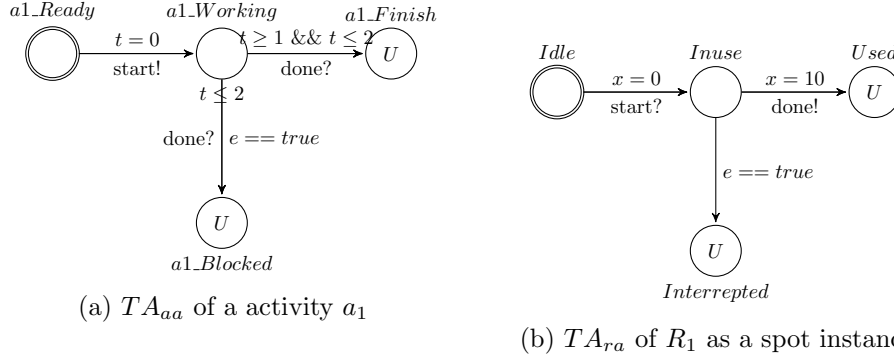


Figure 5.5: Allocation of R_1 as a spot instance with an interruption risk to activity a_1

has five locations: a_{mReady} , $a_{mWorking}$, $a_{mFinish}$, $a_{msuccess}$ and $a_{mfailure}$. Three locations: a_{mReady} , $a_{mWorking}$, and $a_{mFinish}$ are similar to TA_a 's locations. But, $a_{mfailure}$ is reached if the number of activities consuming resource R is greater than its defined capacity (*share* variable). Otherwise, each activity a_m is executed successfully ($a_{msuccess}$).

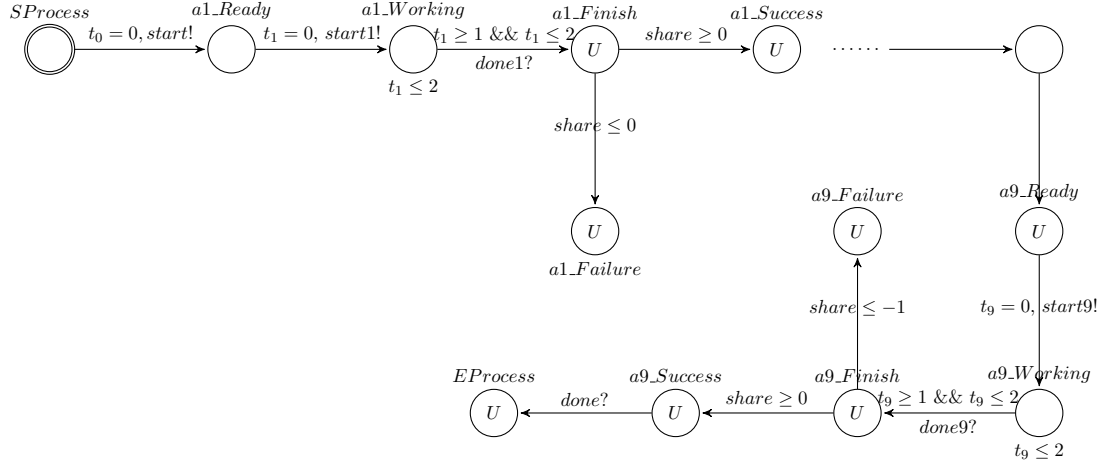
Formally, see Definition 2.5.1, TA_{ash_m} is a tuple where $L = \{a_{mReady}, a_{mWorking}, a_{mFinish}, a_{msuccess}, a_{mfailure}\}$, $l_0 = a_{mReady}$, $X = \{x\}$, $I(a_{mReady}) = \emptyset$, $I(a_{mWorking}) = \{x \leq MaxD_{a_m}\}$, $I(a_{msuccess}) = \emptyset$, $I(a_{mFinish}) = \emptyset$, $I(a_{mfailure}) = \emptyset$, $Tr = \{(a_{mReady}, start_m!, x = 0, \emptyset, a_{mWorking}), (a_{mWorking}, done_m?, x \geq MinD_{a_m} \ \&\& \ x \leq MaxD_{a_m}, a_{mFinish}), (a_{mFinish}, \emptyset, \emptyset, share \geq 0, a_{msuccess}), (a_{mFinish}, \emptyset, \emptyset, share \leq -1, a_{mfailure})\}$.

On the other hand, TA_{rsh} is the timed automata of the shared resource as a reserved or spot block instance. It is composed of $z + 4$ locations: *Available*, *Idle*, *Inuse*, z *Used_m* (where $m \in \{1 \dots z\}$), and *Unavailable*. *Idle* and *Inuse* locations are similar to the ones in TA_{ar} , whereas *Available* means that the resource starts to be available, *Used_m* is reached when R is used by the m th activity, and *Unavailable* means that R can not be used.

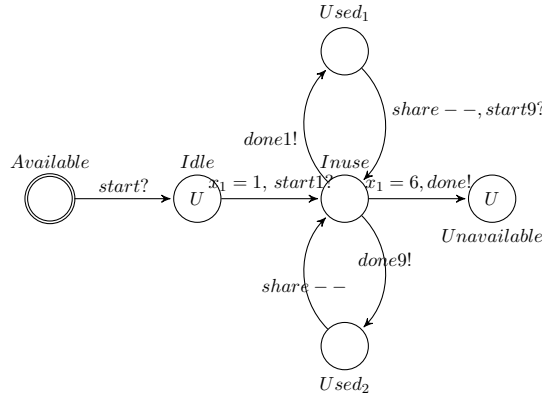
Formally, see Definition 2.5.1, TA_{rsh} is a tuple where $L = Available, Idle, Inuse, Used_m$ (where $m \in \{1 \dots z\}$), *Unavailable*, $l_0 = Available$, $I(Available) = \emptyset$, $I(Idle) = \emptyset$, $I(Inuse) = \emptyset$, $I(Used_m) = \emptyset$, $I(Unavailable) = \emptyset$ and $Tr = \{(Available, start?, Idle), (Idle, start_m?, x = MinAvR, share - -, Inuse), (Inuse, done_m!, x \leq MaxAvR, Used_m), (Used_m, start_{m+1}?, share - -, Inuse), (Used_m, done!, x = MaxAvR, Unavailable)\}$.

For illustration purposes, we consider that $a_1 = (\{Duration((a_1, 1, 2))\}, R_1, \text{spot blocks})$ and $a_9 = (\{Duration((a_9, 1, 2))\}, R_1, \text{spot blocks})$ in the ‘‘supervision process’’ BP. Figure 5.6a presents an excerpt of the supervision process. It shows

the transformation output of a_1 and a_9 . Both activities reach the successful locations if the resource is shared successfully ($share \geq 0$). Otherwise, the failure locations are reached. The timed automata of the shareable resource R_1 is shown in Figure 5.6b. The variable $share$ is used to check if R_1 can be consumed anymore by other activities. It is decremented each time it is used by a new activity.



(a) An excerpt of the supervision BP timed automata: $TA_{a_{sh}}$



(b) $TA_{r_{sh}}$ of R_1 shareable as a spot blocks

Figure 5.6: Allocation of R_1 as a spot block shared between activities a_1 and a_9

5.3.2 Automatic transformation

BPMN processes enriched with pricing strategies help the BP designer to easily allocate cloud resources. However, it may lead to temporal violation in some cases. Therefore, we resort to MDE-based model-to-model transformation language ATL

which is enough mature and reliable to accomplish transformations required in our work. We present one example of ATL code in Listing 5.2. It is an excerpt that shows how we implement the transformation rule of an activity consuming R as a spot block. Figure 5.13d depicts the generated model. Line 2 shows the source element of the BPMN model. Lines 4-7 depict how *Idle* state is created. Similarly, we create *Inuse* and *Used* states [63]. Moreover, we present how the transitions are created in Lines 9-25. For instance, Lines 9-16 depict how the transition from *Idle* location to *Inuse* location. Lines 18-25 present how the transition from *Inuse* location to *Used* location. Lines 27-33 depict how the full timed automata of strategy spot block is composed.

Listing 5.2: ATL code to transform a spot block pricing strategy into timed automata

```

1 rule spotBlock {
2   from spotBL: BPMN!ExtensionAttributeValue (..)
3   to
4   idleLocation: uppaal!Location(
5     id <- 'id' + thisModule.getcounterSpotblocks()
6     name <- thisModule.NewName('Idle'),
7     initLocation: uppaal!Init (ref <- idleLocation.id),...
8
9   transitionIdleInuse: uppaal!Transition (
10    source <- thisModule.SourceTransition(idleLocation),
11    target <- thisModule.TargetTransition(inUseLocation),
12    label <- thisModule.synchronisation('start?'),
13    label <- spotBL.taskConfig.parameterspotBlock ->
14      collect(e |if e.Shared=false then
15        thisModule.assignmentVM('x=' + e.MinAvR)
16        else OclUndefined endif),
17
18    transitionInuseUsed: uppaal!Transition (
19      source <- thisModule.SourceTransition(inUseLocation),
20      target <- thisModule.TargetTransition(usedLocation),
21      label <- thisModule.synchronisation('done!'),
22      label <- spotBL.taskConfig.parameterspotBlock ->
23        collect(e |if e.Shared = false then
24          thisModule.assignmentVM('x=' + e.MaxAvR)
25          else OclUndefined endif),
26
27    templateSpotblocks: uppaal!Template (
28      name <- templateNameSpotblocks,
29      location <- idleLocation,...
30      transition <- transitionIdleInuse,...
31      init <- initLocation),

```

```

32 templateNameSpotblokcs: uppaal!Name (
33 value <- 'templateSpotBlock'})}

```

The output of our automatic transformation is a network of timed automata. So, BP’s designers can check at design time the temporal correctness of cloud resource allocation in BPs without carrying on the prone task of manually generating timed automata model.

5.3.3 UPPAAL Meta-Model

To conduct the model-to-model transformation, it is crucial to present UPPAAL meta-model. But, a standardized meta-model for UPPAAL is still not available. Even the one proposed in [279] is of high complexity and can not cover our needs. Therefore, we propose the meta-model depicted in Figure 5.7 that shows clearly UPPAAL meta-classes and structures. In UPPAAL meta-model, “Nta” is the root meta-class. It is composed of one or more Templates. In “Nta” meta-class, there is one occurrence of system attribute that is used to declare template instantiations and to list one or more processes composed into a system. In “Nta”, we find also declaration attribute to define global declarations of templates’ elements. Otherwise, the declaration attribute that exists in “Template” meta-class is to declare local declarations of a template. These attributes are helpful to declare the different process elements such as synchronization channels, clocks, variables, etc. A template is where a timed automata process is drawn. Each “Template” contains locations linked with transitions. Each “Transition” is composed of one “Source” and one “Target” location by referencing the location’s id. A location can be “Urgent” if necessary. Added to that, every “Location” and “Transition” are composed of a name and one or more “Label” which define any condition related to the temporal constraints. Each occurrence of “Label” meta-class has a value and a kind that can be guard, assignment, invariant, or synchronization. Moreover, “Template” meta-class is composed of zero or one “Init” meta-class which references the initial “Location” id. We mention that there exist x and y attributes in “Location”, “Transition”, “Label”, and “Name” meta-classes. In fact, these attributes refer to the position of the corresponding element on the drawing canvas of UPPAAL.

5.4 Correctness analysis

After transforming a BPMN process model into timed automata model, we proceed with the correctness analysis of these automata. The correctness analysis step provides a model checking method to detect probable temporal violations between allocated cloud resources and BP’s activities. Thus, it enables the BP designer to react to them predictively. The proposed BP verification goes far beyond the simple verification of the structural properties of the model (e.g. deadlock). Precisely, it enables

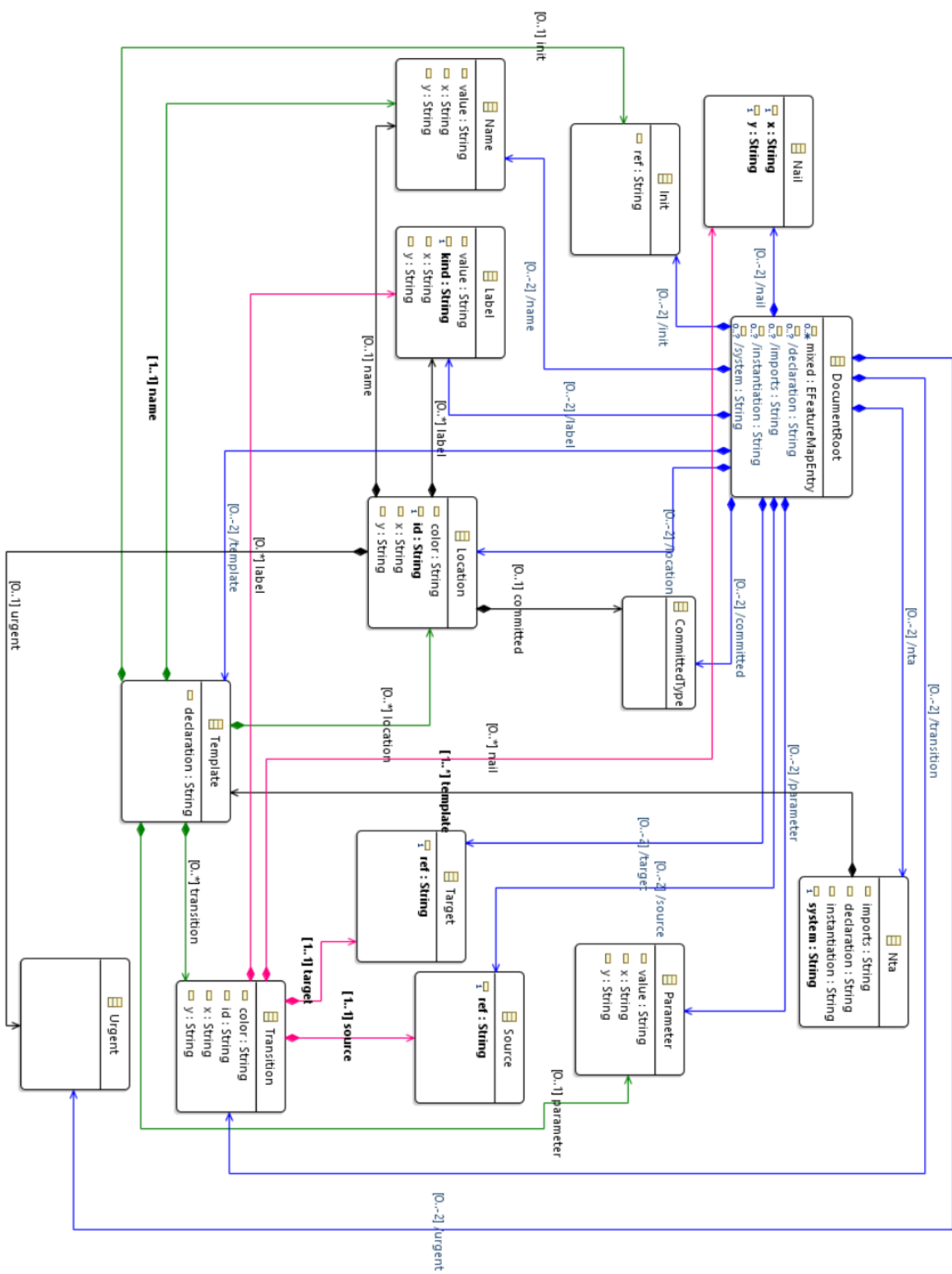


Figure 5.7: UPPAAL meta-model extension

the verification of the advanced temporal constraints and temporal availabilities of cloud resources. We use UPPAAL to check if some properties are satisfied or not, which serves for the verification step after designing the BP. Thus, we help the BP designer to verify the BP temporal correctness of cloud resource allocation. To this end, we check the satisfaction or not of a set of properties such as:

- **Deadlock freeness:** to verify that the system is deadlock free.
- **Liveness:** to verify that an activity consuming a resource will eventually reach the state finish successfully.
- **Deadline:** to verify that the deadline is met.

5.5 Evaluation

Let's consider again the service supervision BP from Orange France Telecom process presented in Section 3.3. Activities have temporal constraints and require computing resources. To evaluate our approach, we assume that a_1 and a_9 consume R_1 as a spot block, a_2 consumes R_2 as a spot instance, a_3 consumes R_1 as on-demand instance, a_4 consumes R_3 as a spot instance, a_5 consumes R_1 as a spot instance, a_6 consumes R_5 as an on-demand instance, a_7 consumes R_3 as an on-demand instance, and a_8 consumes R_2 as spot block.

5.5.1 Supporting pricing strategies description

We have developed a plug-in as a proof of concept to take into account the modeling of time-aware BP deployed in cloud resources proposed under various pricing strategies. This plug-in is an extension of BPMN 2 modeler [141] which is an open source graphical modeling plug-in for BPM supporting the BPMN 2.0 standard on Eclipse, the most widely used integrated development environment. More details about the framework implementation and the plugin are presented in Annexes A and B, respectively

The BP designer uses the extended tool palette to draw the BP (Figure 3.3). He can define activities' temporal constraints as well as cloud resources and pricing strategies. As shown in Figure 5.8 (label 1), the palette is extended with a category Task+CloudResources in which we find an extended Task which highlights that the task flow requires a cloud resource. Besides, the palette is extended with relative temporal constraints category which contains temporal dependency and duration constraints in order to decorate activities. We also extend the palette with absolute temporal constraints Category in which we find MSO, MFO, SNLT, and FNLT.

A BP activity consumes a cloud resource with different pricing strategies as shown in the motivating example (Section 3.3); e.g. a_2 consumes R_2 as a spot instance. As mentioned before, cloud providers offer various pricing strategies under which

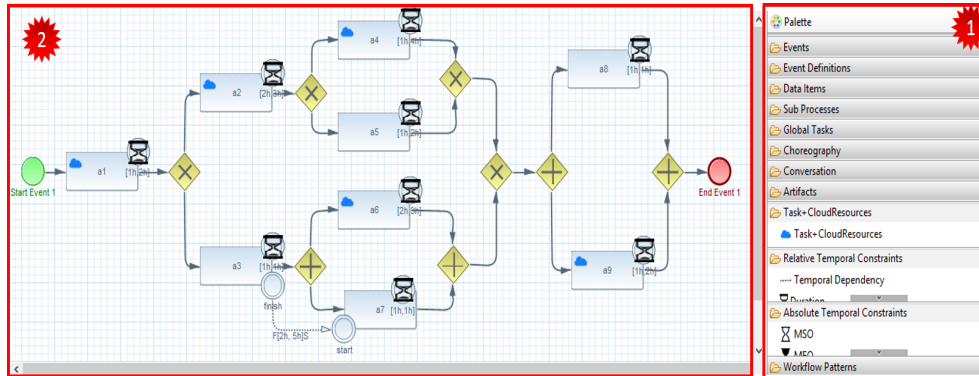


Figure 5.8: Extended BPMN process with BPMN2 modeler

Name	Bid Price Pred	Min AvR	Max AvR	Operating System	Zone	Region	Shared
spotblock1	0.078	1	2	Linux	Oregon	US West	false

Figure 5.9: Pricing strategies specification

resources can be acquired and their allocation costs can be determined. To specify these relations, the BP designer selects the drawing canvas (label 2) in order to display the property view encompassing the set of cloud pricing strategies. Each strategy is in a separate tab as shown in Figure 5.9 (e.g. label 3.1 and 3.2). Once the designer chooses the appropriate pricing strategy, a list and detail widget appear with several control button (mainly add, remove, and edit the selected list item) (label 3).

To assign a pricing strategy, the BP designer should select the activity and press on the “VirtualMachine” property tab. Thus, a small window will appear displaying sections of each pricing strategy and a button under each one to assign a pricing strategy to this activity. The created strategies are added in a combo box in order to assign it to the activity. We give the example of activity a1 in Figure. 5.10.

5.5.2 BPMN model transformation

After modeling graphically the BP in BPMN, it is important to transform it into timed automata. For that, we run our implemented transformation rules. To this end, the BP designer should define a name for the configuration, BPMN and UPPAAL meta-models as well as the source BPMN model and the name of the timed automata for UPPAAL (Figure 5.11). When he runs the transformation rules implemented

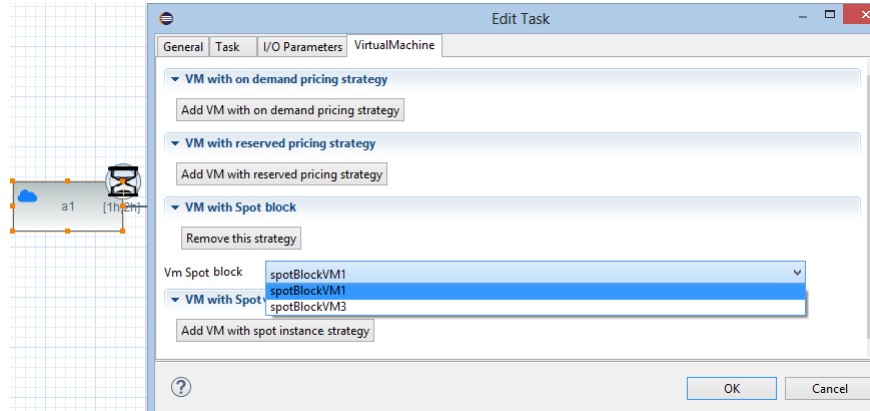


Figure 5.10: Pricing strategies specification

with ATL, an XML file will be generated as an output. Our transformation rules are correct-by-construction since they are neither ambiguous nor conflicting. The generated network of timed automata that the transformation produces are well-formed. These automata networks are consistent with timed-automata meta-model. In addition, the transformation is complete since each element (e.g., activity, gateway, and event) in a time-aware BP, as a source model, has a corresponding element (e.g., state, transition, and guard) in the timed automata, as a target model. An excerpt from this XML file is presented in Listing 5.3. In fact, it shows excerpts of template for the timed automata of BP activities “*ProcessActivity*” (Lines 2-43) and template for spot block strategy “*R2SpotBlock*” (Line 45-70). Each template is composed of locations, transitions, labels, etc.

Listing 5.3: Excerpt from the generated XML file: timed automata for UPPAAL

```

1 <nta xmlns="flat-1.2.dtd" >
2   <template>
3     <name>Process_Activity</name>
4     <location id="id0" >
5       <name>a1.Failure</name>
6     </location>
7     <location id="id2" >
8       <name>EProcess</name>
9       <urgent/>
10    </location>
11    <location id="id27" >
12      <name>a1.Success</name>
13    </location>
14    <location id="id28" >
15      <name>a1.Finish</name>
16      <urgent/>
17    </location>
18    <location id="id29" >
19      <name>a1.Working</name>
20      <label kind="invariant">t1>=2</label>
21    </location>

```

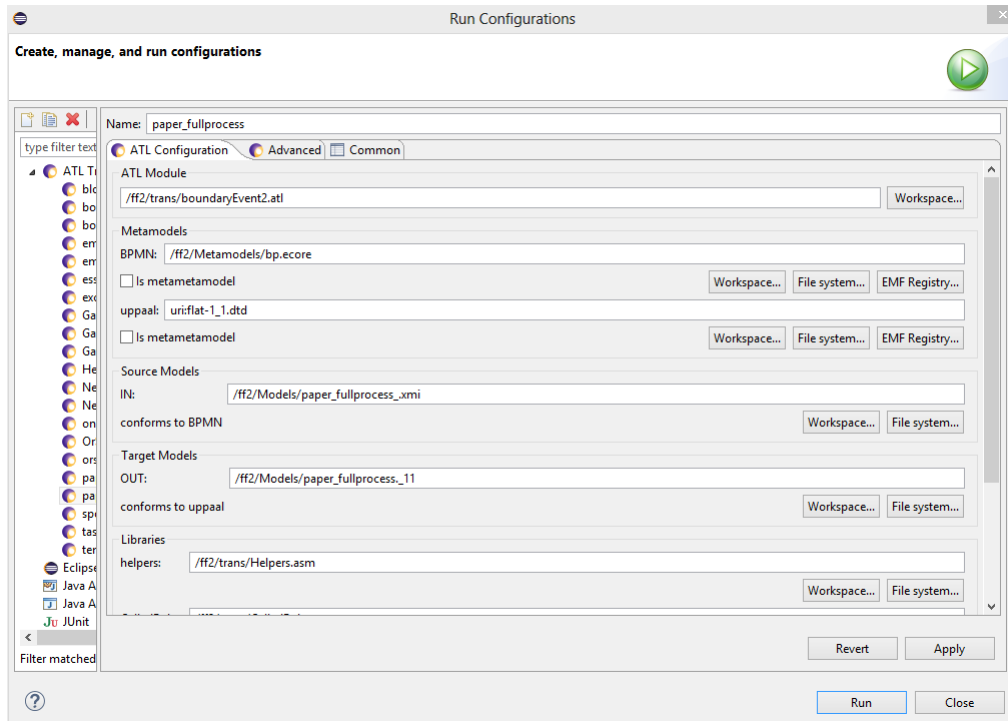


Figure 5.11: ATL transformation

```

22 |         <location id="id30">
23 |             <name>a1.Ready</name>
24 |         </location>
25 |         <location id="id31">
26 |             <name>SProcess</name>
27 |         </location>
28 |         <init ref="id31" />
29 |         ...
30 |         <transition>
31 |             <source ref="id29" />
32 |             <target ref="id28" />
33 |             <label kind="guard">t1>=1 && t1<=2</label>
34 |             <label kind="synchronisation">done1?</label>
35 |         </transition>
36 |         <transition>
37 |             <source ref="id30" />
38 |             <target ref="id29" />
39 |             <label kind="synchronisation">start1!</label>
40 |             <label kind="assignment">t1=0</label>
41 |         </transition>
42 |         ...
43 |     </template>
44 |     ...
45 | <template>
46 |     <name>R2.SpotBlock</name>
47 |     <location id="id41">
48 |         <name>Inuse</name>

```

```

49     </location>
50     <location id="id42">
51         <name>Used</name>
52         <urgent/>
53     </location>
54     <location id="id43">
55         <name>Idle</name>
56     </location>
57     <init ref="id43"/>
58     <transition>
59         <source ref="id41"/>
60         <target ref="id42"/>
61         <label kind="synchronisation">done8!</label>
62         <label kind="assignment">x2=6</label>
63     </transition>
64     <transition>
65         <source ref="id43"/>
66         <target ref="id41"/>
67         <label kind="synchronisation">start8?</label>
68         <label kind="assignment">x2=1</label>
69     </transition>
70 </template>
71 </nta>

```

The BP designer can open this file with UPPAAL model checker to see the generated templates drawn as timed automata (Figures 5.12 and 5.13). Thus, he can proceed to the formal verification of this generated model and check the satisfaction of several properties using CTL formulas.

5.5.3 Checking CTL properties

Model checking is a widely used technique to verify BP models against a wide range of temporal constraints [73]. Our verification approach formally checks the matching between temporal constraints of both activities and cloud resources. In this context, we use UPPAAL for this matching. The BP and cloud resource allocation are modeled as a network of timed automata of : (i) ProcessActivities (activity-timed automata, Figure 5.12) and (ii) Resource (resource-timed automata, Figure 5.13), respectively. Figure 5.12 and Figure 5.13 are composed of locations presenting the activities/resource states, clocks updated in some transitions, guards to control the satisfaction of temporal constraints, invariants to specify the temporal requirement that the system can not stay in a location, and channels variable to ensure the synchronization between both timed automata. For instance, a_2 can start consuming R_2 only if the clock t_2 is less than 3 (i.e., defined as an invariant) and will execute successfully only if its temporal duration is met (i.e., defined as a guard). $a_{2Blocked}$ location is reached when R_2 is interrupted (i.e., defined as a guard). The timed automata of R_2 as spot instance is illustrated in Figure 5.13e. When R_2 is in state *Inuse* the clock x_{28} , taken as a time reference, is initialized to zero. Then, it is initialized to 18 if its finish availability time is reached. The Boolean variable e_2 is “true” to indicate that resource R_2 is interrupted (i.e., defined as a guard). The variables $start_2$ and $done_2$ ensure

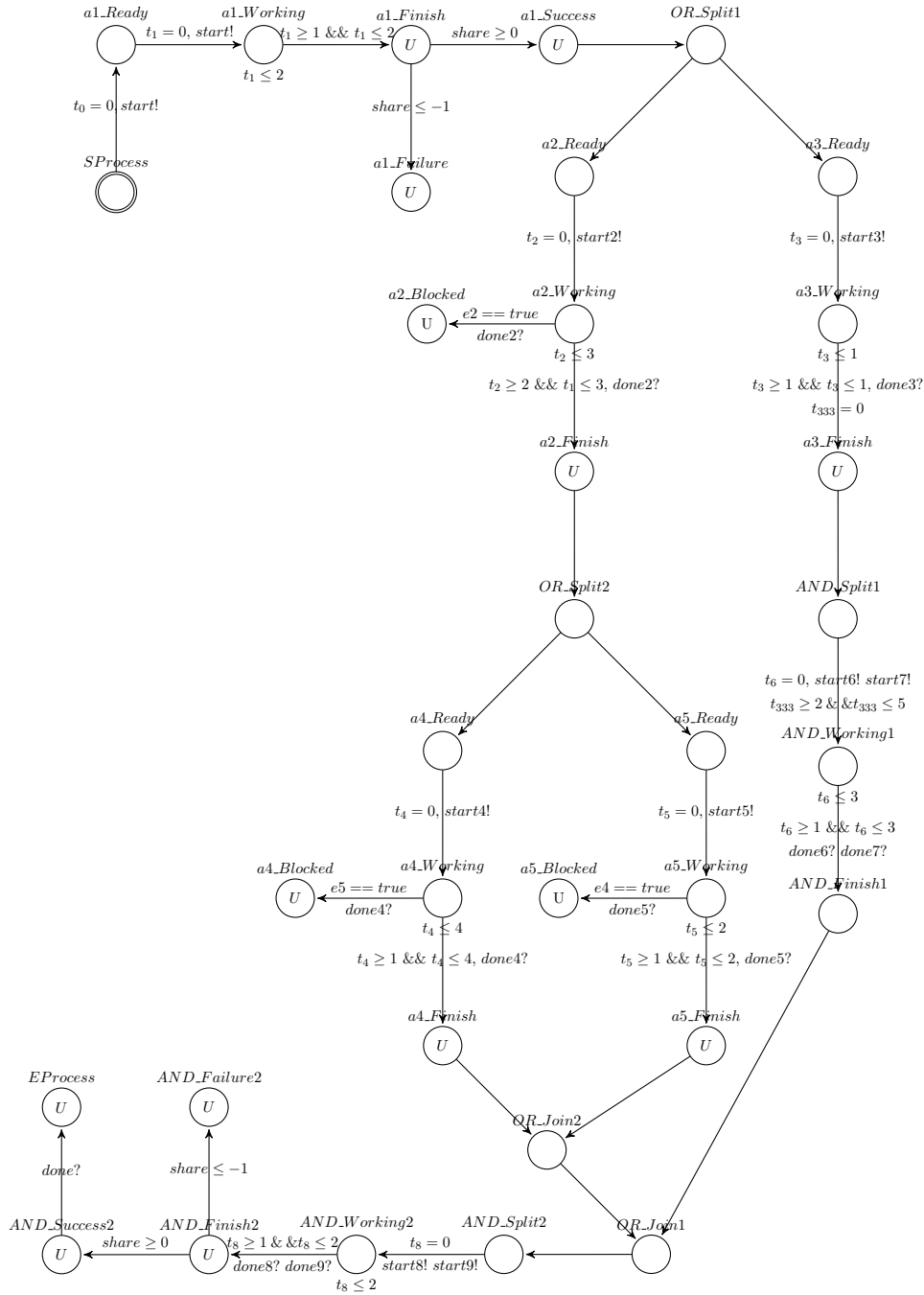


Figure 5.12: Process activities timed automata

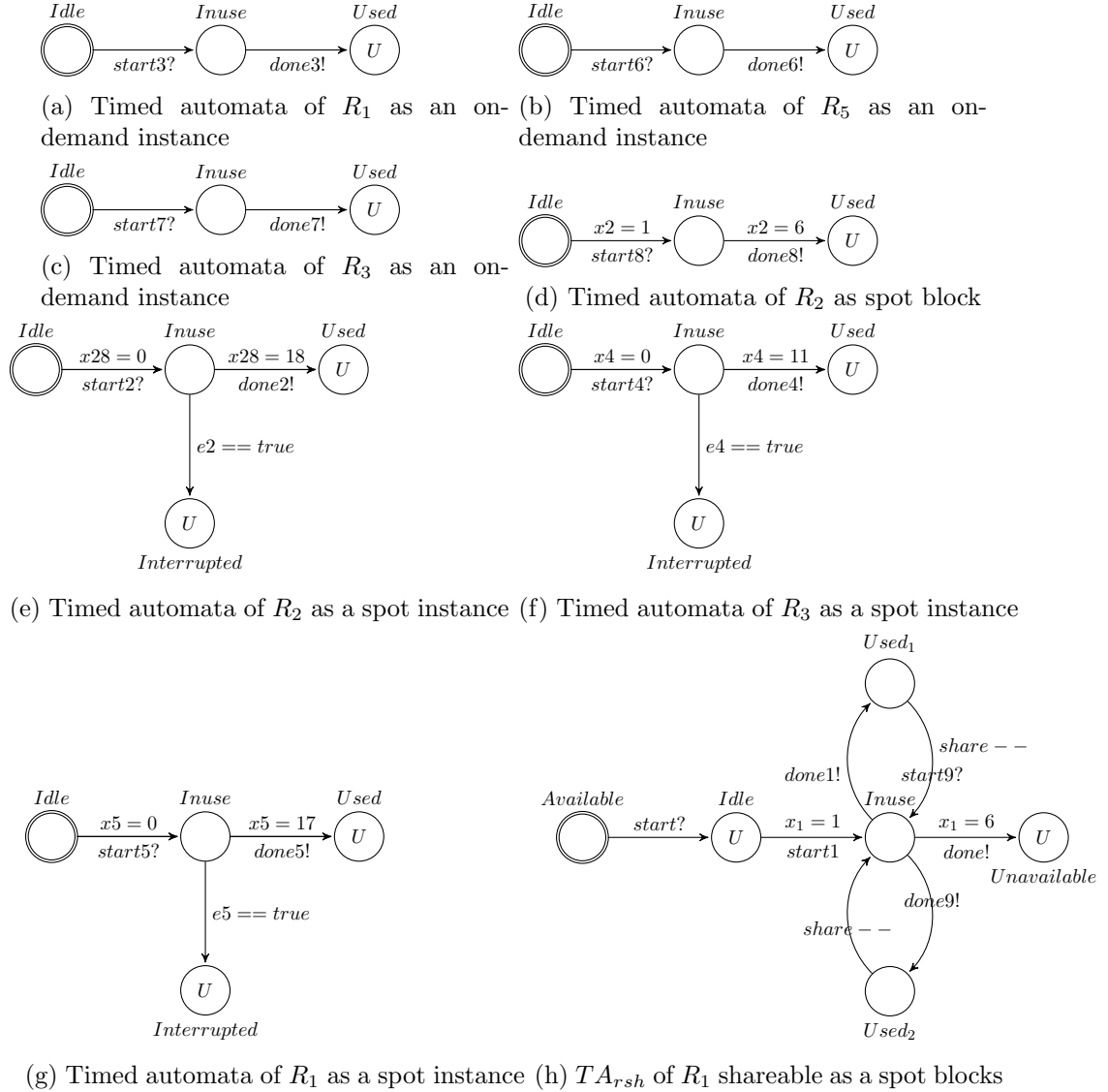


Figure 5.13: Resources timed automata

the synchronization between both timed automata.

Assuming the network of timed automata (Figures 5.12 and 5.13), we use a rich set of CTL formulae to verify some properties such as:

1. **Deadlock:** $A[] \text{ not deadlock}$: the process is deadlock free.
2. **Liveness:** $E<>(ProcessActivities.ANDWorking2 \text{ and } R_2.Inuse \rightarrow ProcessActivities.ANDFinish2)$: if a_8 is consuming R_2 which is still available then even-

tually a_8 will finish successfully its execution.

3. **Deadline:** $A[](\text{ProcessActivities.EProcess imply } t0 \leq 23)$: the process should reach the EProcess state before 23 hours to ensure that its deadline is met.

Activity timed automata, resource timed automata, and formulae are submitted to UPPAAL.

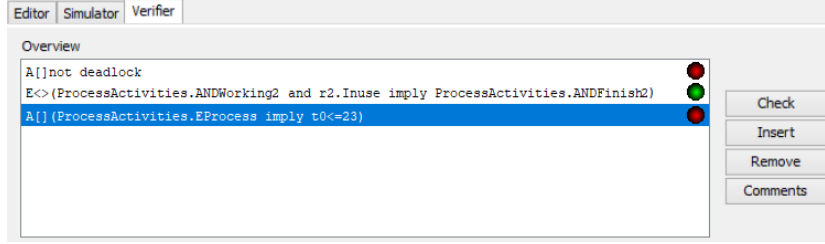


Figure 5.14: UPPAAL verifier's outcomes

Figure 5.14 illustrates UPPAAL verifier window. It shows the verification results of the previous properties. The green light means that the property is satisfied. The red light indicates that the property is unsatisfied. The verification results show that the corresponding resource allocation is not correct, i.e., the process is live, but it is not deadlock free, and does not meet its deadline. Indeed, only the liveness property is satisfied since a_8 is able to finish successfully its execution (i.e., the process can reach ANDFinish2 state). The process is not always deadlock free due to the fact some activities such as a_2 , a_4 , and a_5 , can be blocked due to the interruption of resources consumed as spot instances.

The results provided from UPPAAL demonstrate the usefulness of our solution. Indeed, it helps the BP designer to detect temporal violations resulting from the mismatching between temporal constraints of both activities and cloud resources.

5.6 Conclusion

In this chapter, we answered the two questions mentioned in our thesis problematic (see Section 3.2) which are: *How to support cloud resource allocation in time-aware BP models?*, and *How to ensure the correctness of time-aware BPs in cloud resources?*.

To support cloud resource allocation in time-aware BP models, we proposed a modeling of cloud resources, pricing strategies, as well as time-aware BP. We presented thereafter a BPMN compliant extension to support our proposed model. We particularly presented how the cloud pricing strategies can be specified at design time, through temporal constraints.

To ensure the correctness of time-aware BPs in cloud resources, we propose a set of transformation rules that we implemented using ATL to ensure an automatic generation of timed automata. Consequently, we formally verify the respect of activities'

temporal constraints and cloud resources using model checking. We implemented our approach through a proof of concept demonstrating the feasibility of our proposals.

We note that a set of possible resource allocation options can be provided using our proposed verification step. Each correct allocation provides a different cost. Thus, it is important to find the resource allocation that provides the minimal BP deployment cost. For that, in the following Chapter, we will present our optimization approach to provide the allocation that minimizes the BP deployment cost.

Optimization of Business Process Deployment Cost in Cloud Resources

Contents

6.1	Introduction	121
6.2	Linear optimization	123
6.2.1	Inputs and decision variables	123
6.2.2	Problem constraints	124
6.2.3	BLP	127
6.2.4	MIP	127
6.3	CloudSim simulation	128
6.3.1	CloudSim extension	128
6.3.2	Unified Description Model	129
6.3.3	Simulation of resource allocation	130
6.4	Evaluation	131
6.4.1	Case study	131
6.4.2	Performance Analysis	132
6.4.3	Comparison	136
6.4.4	Impact of the verification step	138
6.4.5	CloudSim Results	141
6.5	Conclusion	143

6.1 Introduction

In this chapter, we present our third contribution that has as objective optimizing BP deployment cost in cloud resources. As mentioned in Chapter 2, cloud providers are diversifying their pricing strategies, specified based on temporal constraints, to

attract more clients and reduce their unused resources' expenditure. So, to minimize enterprises' spending based on less expensive pricing strategies, BPs need to be correctly executed to avoid any temporal violation that could engender serious consequences. Indeed, on-time delivery of goods or services has a direct impact on customer satisfaction. Furthermore, time management is often a very effective cost reduction strategy for organizations [30]. To deal with such issue, the BP designer can resort to our extended version of BPMN, then to our second contribution to ensure the temporal correctness of the resource allocation. After modeling the BP and verifying the non-conflict between temporal constraints (Chapter 5), one of the BP designer challenge is to select for each activity the suitable cloud resource and the best pricing strategy without any constraint violation. In other words, it is important to discover an optimal BP deployment cost while respecting a set of constraints such as time, capacities, etc. However, the variety of cloud resources, pricing strategies, and activities requirements does not help the BP designer to find an optimal BP deployment cost. Some research works have been made in the context of resource allocation in the BPM field [32, 62, 154, 218]. Whereas, others dealt with *optimal* resource assignment and activities' scheduling in cloud [15, 49, 56, 57, 280, 281]. However, they consider the variety of neither pricing strategies nor advanced temporal constraints.

Cloud resources price is variable and timed constrained (e.g., AWS EC2 spot instance). So, the BP designer can waste time, money, and efforts to configure a real cloud environment based on wrong estimations. Therefore, an organization may pay extra fees to deploy its BP correctly. To deal with such issues, researchers often use a simulation tool to model the mechanisms and evaluate the results [28]. To this end, the BP designer needs a cloud simulator that simulates an optimal resource allocation and provides its real cost. In literature, different cloud simulators' extensions are proposed such as TeachCloud [59], CPEE [60], and CloudExp [61]. But, to the best of our knowledge, existing cloud simulators do not provide the optimal and the real BP deployment cost in cloud resources while taking into consideration the variety of pricing strategies.

To overcome the challenges mentioned above, we propose our third contribution composed of two steps: Linear optimization and CloudSim extension. In the first step, we propose an approach that offers two methods to assist the BP designer to find the optimal BP deployment cost. More specifically, we propose, first, a Binary Linear Program (BLP) to provide an assignment method in order to select the suitable cloud resource, cloud provider, and pricing strategy that satisfy activities requirements and resource constraints. Second, in the case of temporal flexibility, we propose a Mixed Integer Programming (MIP) to provide a scheduling method in order to find the start and end times of BP activities to overlap with the temporal availabilities of cloud resources. In the second step, we extend the famous cloud simulator provided in the market, CloudSim [69], to assist the BP designer to simulate a cloud resource allocation in a time-aware BP and to have an estimation of its real cost. This extension is related to the capability to simulate the cloud resources consumed in the BP model.

We combine the proposed extension with an existing one [145] that supports pricing strategies to compute the BP deployment cost and then deduce its real cost.

The contributions presented in this chapter were published in [64–67].

The remainder of this chapter is organized as follows: Section 6.2 details our linear optimization step. Section 6.3 gives an overview about of our simulation step. Computational results are presented in Section 6.4. Finally, Section 6.5 presents our conclusion.

6.2 Linear optimization

In this section, we detailed how the cost of the BP deployment cost is optimized. To this end, we resort to mathematical formulation: BLP and MIP models to define an objective function and constraints that would guide the optimization. Both models are known for their simplicity, flexibility, and extensive modeling capability [282]. We introduce, first, some assumptions and notations to facilitate the mathematical formulation. With no loss of generality, we consider that the BP model has just AND branching and all the activities require cloud resources to be performed. Moreover, we consider that $MaxD_{a_q}=d_q$ and $D_{max}=du$. We assume also that the size of transferred data between cloud resources is not considered and is left as future work. Finally, we assume that the cloud resources are not shareable.

6.2.1 Inputs and decision variables

To begin with, we assume the availability of several cloud resources that satisfy activities' requests of these resources. BLP and MIP take as inputs: A a set of BP activities, \mathcal{R} a set of cloud resources, Pr a set of cloud providers, and St a set of pricing strategies. The set of activities' temporal constraints T_A includes flexible AFT and inflexible $AIiT$ temporal constraints.

The following are the inputs of our BLP and MIP:

- Set of process activities $A=\{a_q : \forall q \in \{1, \dots, z\}\}$, set of activities' requested capacities $Req_A = \{req_{a_q} : \forall q \in \{1, \dots, z\}\}$, and set of activities' temporal constraints $T_A=\{T_{a_q} : \forall q \in \{1, \dots, z\}\}=\{AFT \cup AIiT\}$;
- Set of cloud resources $\mathcal{R} = \{R_i, \forall i \in \{1, \dots, n\}\}$;
- Set of cloud providers $Pr_i = \{pr_{ij}, \forall j \in \{1, \dots, p\}\}$ and set of pricing strategies $St_{ij} = \{st_{ijk}, \forall k \in \{1, \dots, s\}\}$ for each provider pr_{ij} .

In the following, we present the decision variables:

- $X_{ijkq} \in \{0, 1\}$ is the decision variable that assigns a suitable cloud resource, cloud provider, and pricing strategy to an activity. It indicates if activity a_q consumes a resource $R_i \in \mathcal{R}$ that a provider pr_{ij} offers according to a strategy k . It is used in both linear models.

- $V_q \in \{0, 1\}$ is a binary decision variable that indicates if activity a_q uses a spot instance and its penalty price is not null. It is used in both linear models.
- S_{a_q} (respectively F_{a_q}) is a decision variable of type integer used to determine the start (respectively finish) time of each activity $a_q \in A$. S_{a_q} and F_{a_q} represent the flexible absolute temporal constraints of a_q . They are used only in the MIP model.

6.2.2 Problem constraints

The various constraints of the optimization problems are presented below. To explain more each constraint, we use examples from the BP presented in Section 3.3. But, we consider that the BP has only AND Branching type (parallel gateway), i.e., used to model situations where more than one path is executed in parallel and all paths are always executed. It should be noted that, we can also rely on the approach of La Rosa et al. [283, 284] which proposes to map the BP model with ORSplit or XORSplit into various variants with the sequential flow and/or ANDSplit. Therefore, we can, first, generate all the possible process model variants. Second, we can apply our approach for each process variant, separately, even though the process model has different gateways types: AND, OR, XOR.

1. **Execution constraints on activities:** Equations (6.1) and (6.2) guarantee that the resources' capacities in terms of processing and memory should satisfy activities' requirements.

$$\sum_{i=1}^n \sum_{j=1}^p \sum_{k=1}^s \min(RAM_i) X_{ijkq} \geq RAM_{a_q}, \forall q \in \{1, \dots, z\} \quad (6.1)$$

$$\sum_{i=1}^n \sum_{j=1}^p \sum_{k=1}^s \min(vCPU_i) X_{ijkq} \geq CPU_{a_q}, \forall q \in \{1, \dots, z\} \quad (6.2)$$

For instance, activity a_1 can be performed by a resource R_i that can satisfy its requirements. Indeed, the minimal RAM_i and $vCPU_i$ of the selected R_i should be equal or greater to RAM_{a_1} and CPU_{a_1} (Equations (6.1) and (6.2)).

2. **Temporal constraints on activities:** To avoid the violation of BP temporal constraints, we used the following four constraints. For instance, the start time (S_{a_i}) should be after the finish time (F_{a_q}) of all its predecessors (Equation (6.3)). We use Equation (6.4) to ensure the respect of inflexible temporal constraints. Equation 6.3 ensures that each activity's start time (S_{a_q}) is after the maximum end-time (F_{a_o}) of all this activity's predecessors. The time lag between the start and end times founded by our MIP for each activity should respect its temporal

durations (Equation (6.5)). Finally, we use Equation (6.6) to respect the BP deadline t .

$$\max(F_{a_o}) + TD(FS, a_o, a_q, du, du) \leq S_{a_q}, \forall a_q, a_o \in A, o < q \quad (6.3)$$

$$S_{a_q} == MSO_{a_q} \ \& \ F_{a_q} == MFO_{a_q}, \forall q \in \{1, \dots, z\} \quad (6.4)$$

$$0 \leq F_{a_q} - S_{a_q} \leq d_q, \forall q \in \{1, \dots, z\} \quad (6.5)$$

$$S_{a_q} \geq 0 \ \& \ S_{a_q} \leq t \ \& \ F_{a_q} \geq 0 \ \& \ F_{a_q} \leq t, \forall q \in \{1, \dots, z\} \quad (6.6)$$

In Section 3.3, a_2 and a_3 are the successors of a_1 . So, both activities should start after the end of a_1 , i.e, $F_{a_1} \leq S_{a_2}$ and $F_{a_1} \leq S_{a_3}$ (Equation (6.3)). Besides, let us consider that a_1 and a_4 have inflexible temporal constraints ($MSO_{a_1}=8\text{am}$ and $MFO_{a_4}=5\text{pm}$) that should be respected then $S_{a_1} = MSO_{a_1}=8\text{am}$ and $F_{a_4} = MFO_{a_4}=5\text{pm}$ (Equation (6.4)). Further, while the duration of activity a_2 is $d_2=3$ hours then the time lag between its start time S_{a_2} and its finish time F_{a_2} shall be between d_2 (Equation (6.5)). Finally, the BP has a deadline constraint $t=24$. Therefore, all the start and finish times of BP activities should be included in the temporal interval $[0,24]$ (Equation (6.6)).

3. **Temporal constraints on pricing strategies:** Equations (6.7) and (6.8) ensure that the temporal duration of a cloud resource R_i should be greater or equal to the temporal duration of the activity requiring it. Moreover, R_i should be available from the start until the end times of the activity consuming it (Equations (6.9), (6.10)).

$$\sum_{i=1}^n \sum_{j=1}^p \sum_{k=1}^s \text{MinAv}R_i X_{ijkq} \geq d_q, \forall q \in \{1, \dots, z\} \quad (6.7)$$

$$\sum_{i=1}^n \sum_{j=1}^p \sum_{k=1}^s \text{MaxAv}R_i X_{ijkq} \geq d_q, \forall q \in \{1, \dots, z\} \quad (6.8)$$

$$\sum_{i=1}^n \sum_{j=1}^p \sum_{k=1}^s \text{SUNET}(R_i) X_{ijkq} \leq S_{a_q}, \forall q \in \{1, \dots, z\} \quad (6.9)$$

$$\sum_{i=1}^n \sum_{j=1}^p \sum_{k=1}^s \text{FUNET}(R_i) X_{ijkq} \geq F_{a_q}, \forall q \in \{1, \dots, z\} \quad (6.10)$$

Each activity has temporal constraints that should meet the temporal constraints of pricing strategies (Equations (6.7) and (6.8)). For instance, $\text{Max}D_{a_1}=d_1=2$ hours is the maximum duration of a_1 . R_i is a spot block can be assigned to a_1 . Thus, $\text{MinAv}R_i \geq d_1$ and $\text{MaxAv}R_i \geq d_1$. When defining activities start and end times, one has to select the resource, the provider, and the pricing

strategy having the cheapest cost while ensuring the respect of pricing strategies constraints. For example, R_1 from pr_{11} in strategy $st_{114}=(\text{spot}, [1\text{am}, 6\text{pm}], 0.0386\$)$ will be allocated for a_1 , then $S_{a_1} \geq SUNET(R_1) = 1\text{am}$ and $F_{a_1} \leq FUNET(R_1) = 6\text{pm}$ (Equations(6.9) and (6.10)).

4. **Constraint interruption:** This constraint is used to ensure the addition of the penalty price when the selected instance has an interruption risk (i.e., spot instance $str_k = 1$).

$$\sum_{i=1}^n \sum_{j=1}^p \sum_{k=1}^s X_{ijkq} str_k = V_q \text{ where } p_q > 0 \text{ and } str_k = 1, \forall q \in \{1, \dots, z\} \quad (6.11)$$

The BP presented as a motivating example in Section 3.3 has some activities subject to financial penalties price when they are canceled due to resource interruption. For example, activity a_4 penalty cost is equal to $p_4=0.2\$$. Thus, if a_4 consumes a cloud resource as a spot instance, (Equation (6.11)) so p_4 should be added to the process total cost.

5. **Assignment Constraint:** Equation (6.12) ensures that one resource offered by one cloud provider under one pricing strategy is used by one activity.

$$\sum_{q=1}^z X_{ijkq} = 1, \forall i \in \{1, \dots, n\}, \forall j \in \{1, \dots, p\}, \forall k \in \{1, \dots, s\} \quad (6.12)$$

6. **Placement constraint:** This constraint is used to guarantee that each activity consumes only one resource proposed by one cloud provider under a specific pricing strategy.

$$\sum_{i=1}^n \sum_{j=1}^p \sum_{k=1}^s X_{ijkq} = 1, \forall q \in \{1, \dots, z\} \quad (6.13)$$

For instance, a_1 can use only one cloud resource R_1 from $pr_{11}=\text{Amazon}$ under $st_{113}=\text{spot}$ strategy. This R_1 performs only a_1 then $X_{1131}=1$ (Equations (6.13) and (6.12)).

7. **Binary constraints:** to ensure that our linear models are binary, we impose that the decision variables should be either 0 or 1 (Equations 6.14 and 6.15).

$$X_{ijkq} \in \{0, 1\}, \quad \forall q \in \{1, \dots, z\}, i \in \{1, \dots, n\}, j \in \{1, \dots, p\}, k \in \{1, \dots, s\} \quad (6.14)$$

$$V_q \in \{0, 1\}, \quad \forall q \in \{1, \dots, z\} \quad (6.15)$$

The total execution cost C (Equation (6.16)) includes two terms. The first is the sum of resources allocation costs in order to execute the BP. The allocation cost is given

$$C = \sum_{i=1}^n \sum_{j=1}^p \sum_{k=1}^s \sum_{q=1}^z d_q c_{ijk} X_{ijkq} + \sum_{q=1}^z p_q V_q \quad (6.16)$$

by the multiplication of the execution time of the activity d_q by the R_i utilization cost. The price of resource R_i is the hourly unit strategy price c_{ijk} (Definition 5.2.2) proposed by a provider pr_{ij} . The second term is the penalty price p_q values added when activities subject to financial penalties are performed by spot instances.

$$\text{Min}C \quad (6.17)$$

The objective function (Equation (6.17)) of the model minimizes the BP deployment cost in cloud resources. Each linear model is subject to a set of constraints presented above.

6.2.3 BLP

Using our BLP model, the objective function (Equation (6.17)), subject to the constraints (Equations (6.1)-(6.3), and (6.7)-(6.15)), seeks to: (i) select for each activity the cloud resource that has the required capacities and (ii) select the suitable pricing strategy for each cloud resource. More precisely, it selects resources that respect the constraints in order to find the best resource assignment that achieves the minimal total BP deployment cost. Consequently, each activity will consume the resource that satisfies its requirements in terms of RAM and processing. Moreover, the pricing strategy selected for each resource should respect activities temporal constraints and would have an impact on the BP deployment cost. In fact, assigning a cloud resource as a spot instance (i.e has an interruption risk) to an activity subject to financial penalty price will lead to increase the total cost. Thus, the addition of penalty price should be reduced to optimize the BP deployment cost. It should be noted that, in our BLP model, S_{a_q} and F_{a_q} are not decision variables but they are considered as inputs. They specify the start and end times of each activity a_q .

6.2.4 MIP

We define our MIP model based on an extension to the constraints of our BLP model. Indeed, to define our MIP, we add for our BLP Equations (6.4), (6.6), and (6.5) to ensure that the scheduling solution respects the temporal constraints of activities. Similarly to our BLP, the objective function of our MIP (Equation (6.17)), attempts to: (i) select for each activity the cloud resource that has the needed capacities and (ii) select the suitable pricing strategy for each cloud resource and in case of temporal flexibility. However, it differs from the objective function of our BLP in two important

ways. First, it is subject to (Equations 6.3-6.14), and second it attempts also to (iii) define the start and end times of each BP activity to overlap with the cheaper pricing strategies temporal constraints. More specifically, it provides the optimal total deployment cost by selecting resources that satisfy the constraints and determining activities' start and end times to find the best scheduling. So, the scheduling execution plan defines for each activity its temporal execution period and the cloud resource that satisfies its requirements in terms of RAM and processing. Moreover, as mentioned in the previous section, our objective is to minimize the penalty price added in case of a resource interruption due to the spot instance bidding process. Thus, we need to avoid the selection of resource with an interruption risk for activities subject to high financial penalties prices.

6.3 CloudSim simulation

Testing in real world environment is one technique for evaluating the performances of a system. However, cloud resources price is variable and timed constrained (e.g., spot instance), so this technique becomes more expensive and time consuming method. Thus, with a simulation tool, the BP designer can consider many different scenarios and get a more real cost. In this section, we describe our Simulation step using CloudSim. We describe, first, how CloudSim has been extended. Second, we give a description of the Unified Description Model. Third, we detail how to simulate the cloud resources consumed in a time-aware BP model in order to deduce the real deployment cost using the CloudSim extension [66].

6.3.1 CloudSim extension

In Occiware project¹, in which Telecom SudParis is partner, a CloudSim extension, *PriceCloudSim* [145], was proposed to support the pricing strategies of Amazon Web Services (AWS). *PriceCloudSim* proposes four different prices for each allocated resource based on on-demand, reserved, and spot block, and spot instance strategies. The cost estimation is done based on three phases. First, the cloud architect models graphically the cloud configurations using the simulation designer, which is a graphical toolkit that helps the architect to easily define the input of the simulator. Second, *PriceCloudSim* simulates the cloud configuration to estimate the execution time of each cloudlet (i.e., cloud activity). Third, *PriceCloudSim* uses the AWS API to retrieve the real price with the different strategies of the resources required for the time estimated. *PriceCloudSim* proposes the same prices given by AWS calculator², which is a useful tool that estimates the cost of cloud resources before the deployment and determines the best and worst-case scenarios.

¹<https://www.occware.org/>

²<http://aws.amazon.com/calculator>

However, *PriceCloudSim* [145] and also AWS calculator did not support the simulation of cloud resource allocation in the context of time-aware BPs. To fill this gap, we extend *PriceCloudSim* [66] in order to simulate the execution of all activities based on the BP control-flow. Each BP activity is considered as an application (Cloudlet in CloudSim) being hosted in the virtual machine and all VMs are deployed in the same Host. Three cases are identified:

1. *Sequential activities*: the execution of one activity follows the execution of another activity. This pattern is represented in CloudSim by SpaceShare policy (see Section 2.3.3).
2. *Conditional branching*: only the branches that satisfy the predetermined criteria are executed. Depending on the number of branches chosen, this pattern can be represented in CloudSim by TimeShare or SpaceShare policies.
3. *Concurrent branching*: all activities in the same branch are executed concurrently. This pattern is represented in CloudSim by TimeShare policy (see Section 2.3.3).

The BP activities are constrained by relative and absolute temporal constraints. For that, we extend *PriceCloudSim* to respect the defined constraints when simulating the allocation of cloud resources. That means that each activity that requires cloud resources needs to be executed at a specific time or during a time interval. The BP, the execution time of each activity, and the allocated cloud resources are defined in the UDM as presented in Section 6.3.2. This UDM is parsed to create a BP instance. This instance along with its allocated resources are then simulated in the *PriceCloudSim*. To support the temporal constraints, CloudSim Cloudlet and Broker classes (see Section 2.3.3) has been extended. The Cloudlet class is extended by new attributes (such as `startTime` and `endTime` attributes) and methods for choosing the ideal moment of the Cloudlet execution. While, the Broker class is extended by delaying the simulation time of the current activity, from the previous one.

6.3.2 Unified Description Model

The creation of tests on real infrastructure involving frequent changes is a time consuming task. Thus, to bridge this gap, we propose an UDM that allows to explicitly model the cloud resources, pricing strategies, and activities temporal constraints in context of time-aware BP. In this section, we present the UDM which describes the BP control-flow, its requirements and the allocated cloud resources. The UDM is defined as an XML document and it is composed of a set of tasks and resources. Listing 6.1 shows an extract of the UDM model of the BP presented in Figure. 3.3.

The **Tasks** correspond to the activities of the BP. Each activity has a set of attributes, temporal dependencies with other activities, and the required cloud resources. The attributes identify and describe the temporal duration. The temporal

dependencies express the relationship between the activities through an operator. The resource identifies the cloud resource needed by the activity. For instance, lines 4-10 in Listing 6.1 depict that the task a_1 starts at 2017-06-02 14:08:09. The minimum duration is 1 hour and the maximum duration is 2 hours. During this time, this task requires VM_1 as a resource. When a_1 was finished, both a_2 and a_3 start in parallel.

The **Resources** describe the set of resources used by the CloudSim tool. In this paper, we consider only the computing (i.e., virtual machine) resources. However, other resources can be attached to tasks such as storage and network. For instance, lines 19 in Listing 6.1 shows the definition of the resource VM_1 and its characteristics. The type of this VM is *x.large* (the number of the virtual CPU is 4 and the memory amount is 16 GB) and its operating system is Linux. The proposed price is defined using on-demand pricing strategy and it is located on North of Virginia.

Listing 6.1: Unified Description Model based on Figure. 3.3

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <Process>
3   <Tasks>
4     <Task id="1" minDur="1" maxDur="2" StartTime="2017-06-02 14:08:09" endTime="
      2017-06-02 14:08:09" length="10000">
5       <attachedTo op="Parallel">
6         <target id="2" startAfter="2" />
7         <target id="3" startAfter="3" />
8       </attachedTo>
9       <Resource>VM.1</Resource>
10    </Task>
11    <Task id="2" minDur="2" maxDur="3" StartTime="2017-06-02 14:09:29" endTime="
      2017-06-02 14:08:09" length="10">
12      <attachedTo op="Sequentiel">
13        <target id="4" startAfter="2" />
14      </attachedTo>
15      <Resource>VM.2</Resource>
16    </Task>
17    ...
18  </Tasks>
19  <Resources>
20    <Resource id="VM.1" name="VM" type="xlarge" location="US East (N. Virginia)" strategy="on
      -demand" os="Linux" dcpu="4" memory="16"/>
21    <Resource id="VM.2" name="VM" type="xlarge" location="US East (N. Virginia)" strategy="
      Reserved" os="Linux" dcpu="2" memory="15"/>
22    ...
23  </Resources>
24 </Process>

```

6.3.3 Simulation of resource allocation

In this doctoral research, we take as inputs: the time-aware BP, and the cloud resources and pricing strategies selected using our BLP and MIP models. Then, we use our Eclipse-plugin (Chapter 5) to design those inputs as a BPMN model in order to obtain the UDM file, i.e., XML document. The latter contains the BP control-flow, requirements, and the allocated cloud resources. Then, using the temporal constraint

defined in this UDM and the pricing strategies defined in *EPriceCloudSim*, we simulate a cloud resource allocation in a time-aware BP to estimate the real cost of the BP deployment.

6.4 Evaluation

In this section, we evaluate our proposal for optimal BP deployment cost in cloud resources to demonstrate its effectiveness, feasibility, performance, and scalability. We first evaluate the feasibility through a case study. Second, we study the impact of different parameters such as penalties price, process structure variation, inflexible temporal constraints, and deadline constraints (Equation (6.4)). Third, we compare the results given by our assignment solution (BLP), our scheduling solution (MIP), and our simulation step. Thus, using the extension of CloudSim simulator to support pricing strategies (on-demand, reserved, spot instance, and spot block), proposed in [285], we simulate the resource allocations provided by our BLP and MIP models to compute the BP deployment costs using updated cost values. Moreover, we compare the performance of our solutions against a resource allocation method. In the end, we demonstrate the scalability of our proposal through studying the impact of our verification step. For the evaluation, we implemented the proposed linear models using IBM-ILOG Cplex Optimization Studio V12.6.3 on a laptop with a 64-bit Intel Core 2.3 GHz CPU, 6 Go RAM, and Windows 10 as OS.

6.4.1 Case study

The section below evaluates the feasibility of our proposals. To this end, we take as inputs: the “service supervision” BP (Section 3.3) with only AND Branching types, and the set of cloud resources presented in Tables 3.2 and 3.3. Further, we assume that the BP has only AND Branching type and must start at 00 am and must finish on the same day. As was mentioned in Section 6.2.2, using the approach of La Rosa et al. [283, 284], we can map the BP model with ORSplit or XORSplits into various variants with the sequential flow and/or ANDSplit. Then, we can apply our approach for each process variant, separately, even though the BP model has different gateways types: AND, OR, XOR.

We present in Table 6.1 the assignment and the BP deployment cost provided through our BLP model which selects for each activity a cloud resource and a pricing strategy. For instance, a_1 starts at 00 am and consumes R_1Spot . The objective function value is 4.016\$.

We present in Figure. 6.1 the scheduling plan of the BP (Figure. 3.3) presented as a motivating example. The execution periods are depicted as green rectangles with a tag on it defining the activity name and the allocated resource is mentioned in red colour. The objective function provided by our MIP model is 3.45\$. While Table 6.2 presents

Table 6.1: Assignment result

Activities	Start time	Finish time	Instance
a_1	00am	2am	$R_1SpotBl$
a_2	2am	5am	$R_2SpotBl$
a_3	2am	3am	R_1Res
a_4	5am	9am	$R_3SpotBl$
a_5	6am	8am	R_1Spot
a_6	3am	6am	R_5
a_7	3am	4am	R_3Res
a_8	9am	10am	R_4
a_9	9am	11am	R_2Spot
Objective function	4.016\$		

the activities execution order and the price of each allocated resource. The cost estimated by the extended CloudSim tool is about 3.81\$.

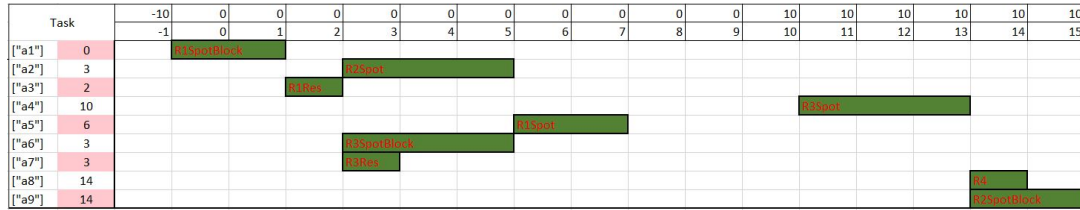


Figure 6.1: Gantt chart of the service supervision process

We note that our MIP gives a cheaper cost compared to BLP and CloudSim values. Whereas, we note that our MIP gives greater end time value (4 pm). This is because some temporal constraints of the BP are flexible and can be delayed or advanced. We conclude that the temporal flexibility helps to reduce the BP deployment cost. Indeed, advancing or delaying the start and/or end times of activities to overlap with resources' temporal availabilities saves up to 14%.

6.4.2 Performance Analysis

In this section, we begin by presenting the data inputs used in our evaluation. Then, we examine the impact of penalties prices, then AND Split/Join, after the temporal flexibility, and finally the deadline constraints on the objective function and the computation time of our linear models.

Table 6.2: CloudSim cost estimation

Activities	RAM	vCPU	Instances	Strategy	Price
a1	16GB	4	t2.xlarge	Spot	[0.0557, 0,1114]\$
a3	17GB	2	m2.xlarge	Reserved (No Upfront)	0.1110\$
a2	15GB	2	r3.large	SpotNo	0.0323\$
a6	30GB	8	m3.2xlarge	Spot	[0,2468, 0,3702]\$
a7	30GB	8	m3.2xlarge	Reserved (No Upfront)	0.3800\$
a5	16GB	4	t2.xlarge	SpotNo	0.0557\$
a4	30GB	8	m3.2xlarge	SpotNo	0.1238\$
a8	15GB	2	DC4s (Microsoft)	On-demand	0.3950\$
a9	15GB	2	m2.xlarge	Spot	[0.0245, 0,049]\$

Table 6.3: Data Input Ranges

Information	Type	Range
Providers' number	integer	[1, \dots , 2]
Amazon strategies' number	integer	[1, \dots , 4]
Microsoft strategies' number	integer	1
vCPU number	integer	[2, \dots , 8]
RAM amount	double	[15, \dots , 64]
Compute price	double	[0.01\$, \dots , 0.532\$]
Requirement in CPU	integer	[2, \dots , 8]
Requirement in RAM	double	[15, \dots , 64]
Activities' number	integer	[2, \dots , 90]
Activities' durations	integer	[1, \dots , 5]
Penalty cost	double	[0\$, \dots , 1\$]

6.4.2.1 Data inputs

Our work is motivated through a real use case from France Telecom/Orange labs. In fact, we did not find a real dataset available to use as input to evaluate our proposal. This is due to the fact that organizations consider mostly such datasets, enriched with cloud resources and pricing informations, as confidential and can not be shared with others. Thus, we relied on our use case requirement values to define the ranges bounds for synthetic datasets generation. In other words, the lower and higher bounds are specified based on real data. For this reason, the data inputs, used in our work, are defined randomly from the real ranges values presented in Table 6.3. For instance, the number of providers should be in (1,2).

6.4.2.2 Penalties Price Evaluation

To reduce the penalties prices due to the interruption risk of critical activities, we use the penalty constraint (Equation (6.11)). As a result, penalty prices may have

an important impact on the BP deployment cost. Thus, we compare the values of the objective function and the computation times of our BLP and MIP models when: (i) all penalty prices are null and (ii) some activities penalty prices are not null. For this end, we modify the number of BP activities and cloud resources proposed by providers under various pricing strategies.

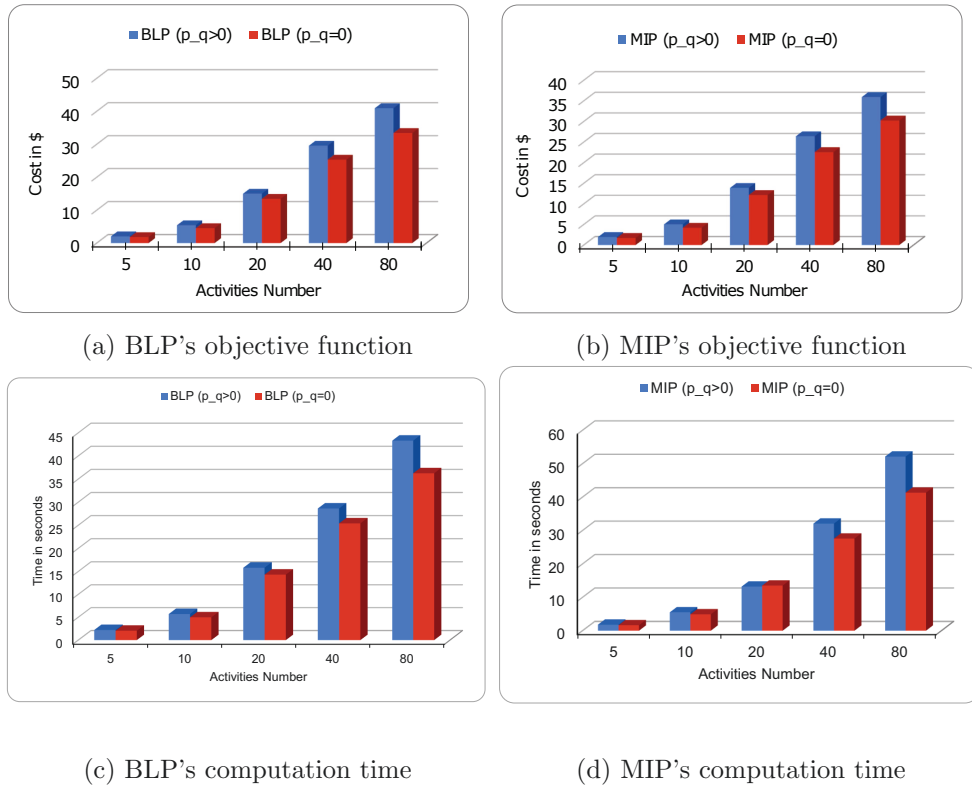


Figure 6.2: Penalties prices

From data shown in Figure 6.2, it is apparent that the objective function and the computation time values are greater when activities have penalty prices for both models (BLP and MIP).

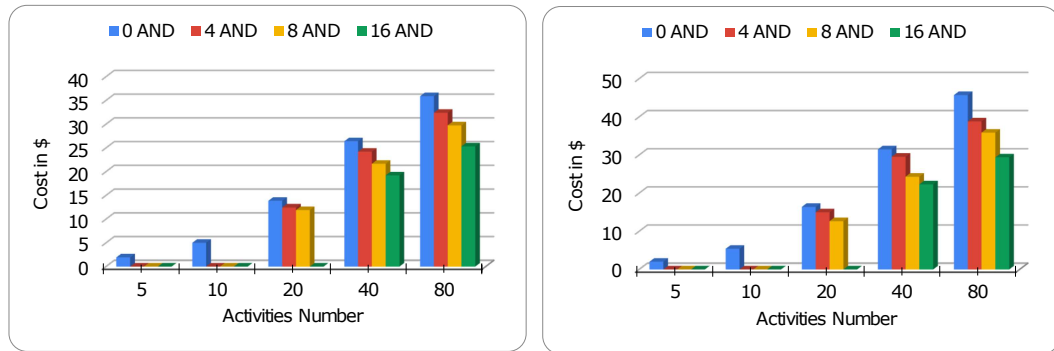
This result may be explained by the fact that the penalty price constraint limits the search space and gives higher values. First, avoiding the selection of instances with an interruption risk for critical activities may lead to select a more expensive resource (i.e., without an interruption risk). Thus, the BP deployment cost is more expensive. Besides, providing the optimal solution takes more time while penalty constraint (Equation (6.11)) restricts more the search space. Consequently, both, objective function and computation time, values increase when some activities have penalties prices. It can, therefore, be assumed that an organization can save up to 20% in BP deployment cost and can reduce up to 22% in computation time to obtain

the optimal solution when activities penalty prices are null.

6.4.2.3 AND Split/Join Constraints

The BP structure is one of the factors that may have an impact on BP’s deployment cost. In this work, we assume that a BP has only ANDSplit branching. Therefore, we vary the number of this branching type on the BP models to evaluate the objective function values provided by both linear models (BLP and MIP).

Figure. 6.3 shows the results obtained from ANDSplit branching variation. We notice that the BP deployment cost is greater when AND ’s number is lesser. In general, therefore, it seems that the parallelism helps to reduce the objective function value. More precisely, the parallelism gives more flexibility for the selection of cloud resources. Indeed, it permits to modify the execution order of some activities to be performed at a cheaper cost. Consequently, we conclude that the BP deployment cost depends on its structure.



(a) BLP’s objective function

(b) MIP’s objective function

Figure 6.3: AND Split/Join Variation

6.4.2.4 Temporal Flexibility Constraint

One of the important factors in our work is temporal flexibility that should be considered to decide which linear model may be used to find an optimal BP deployment cost. The rate of temporal flexibility may have an impact on the MIP objective function value. For this purpose, we vary the rate of temporal flexibility and we evaluate the outputs of our MIP.

From the graph below (Figure. 6.4) we can see that the objective function value increases when the rate of inflexible temporal constraints increases. More specifically, the highest value is obtained when the rate of flexible temporal constraints is 25%. Namely, when 75% of start and finish times of activities are fixed, our MIP does not

have a large flexibility to provide optimal and cheaper BP's scheduling. This led to increase the BP deployment cost while those temporal constraints should be respected. Namely, the presented analysis raises the possibility that the temporal flexibility rate helps to minimize the BP deployment cost.

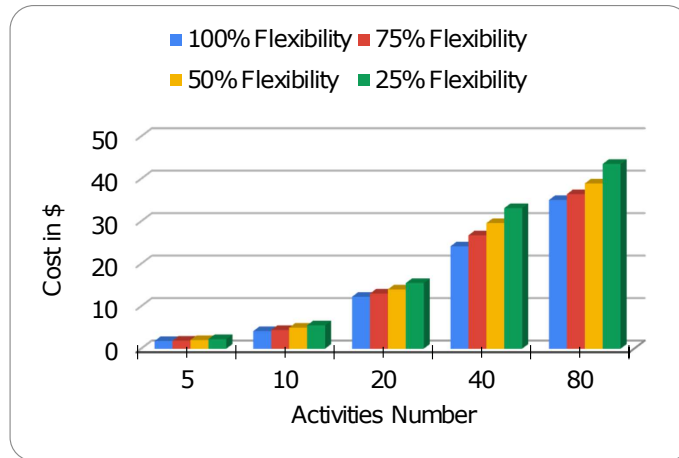


Figure 6.4: Flexibility Evaluation

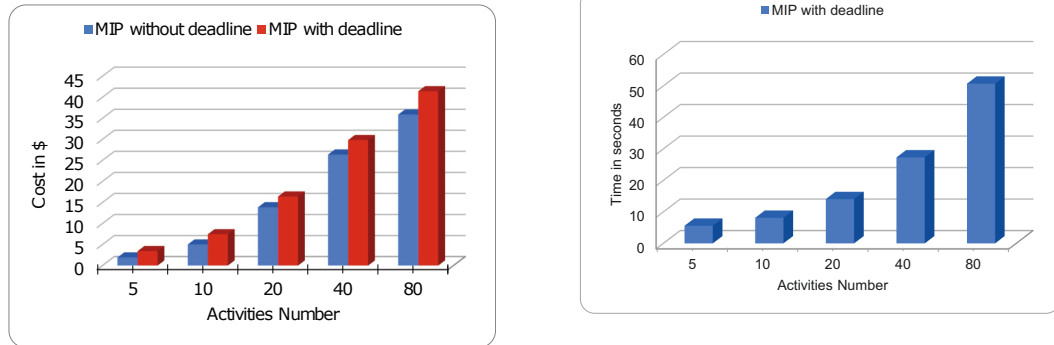
6.4.2.5 Deadline Constraint

In our MIP, we use a deadline constraint (Equation (6.6)) to limit the activities start and end times. Thus, we intend to investigate the objective function and computation time values when this constraint is removed. For that, we compare the results provided by our MIP model with and without deadline constraint.

As shown in Figure. 6.5a, we can notice that the objective function provided by our MIP model is higher under deadline constraint. In Figure. 6.5b, we present only the computation time values when our MIP model is subject to the deadline constraint. By contrast, the computation times have high values (high number of hours) when the deadline constraint is removed. As a result, the computation time is considered as a good indicator of the importance of Equation (6.6). A possible explanation for both results might be that the deadline constraint restricts the search domain of our MIP model. As a result, the deadline constraint gives higher values in a limited computation time. Finally, we conclude that the deadline constraint may help our MIP model to reduce its computation time to provide optimal solutions.

6.4.3 Comparison

In this experiment, we make, first, a comparison between our optimization methods. Second, we compare both of them to another method which is a priority-based



(a) MIP's objective function

(b) MIP's computation time

Figure 6.5: Deadline evaluation

scheduling algorithm (“Priority+FCFS” [93]). There exist different research works that focused on optimizing different parameters such as energy, security, cost, etc in the cloud computing context. Some of these works used well-known scheduling algorithms such as First Come First Served (FCFS) to evaluate their proposed approaches. FCFS is a simple scheduling algorithm that automatically executes queued requests and processes in order of their arrival. So, the authors adapted FCFS to their context to evaluate the comparability. Moreover, to the best of our knowledge, there is no work done around the optimization of the deployment cost of a time-aware BP in cloud resources based on pricing strategies. Consequently, it is not feasible to compare between other existing works and our new approach. Therefore, we adapt FCFS to our context to evaluate our approach.

The priority scheduling algorithm is a non-preemptive algorithm and one of the most common scheduling algorithms. Its basic idea is straightforward: a priority is assigned for each activity. Activity with the highest priority is to be executed first and so on. Equal-Priority activities are scheduled in First Come First Served (FCFS) order [93]. Priority can be decided based on different factors such as memory requirements, time requirements or any other resource requirement. In our case, priority is based on the process control flow. In other words, the highest priority is given to the first activity and the lowest one is given to the last activity. We assign, in the decreasing priority order, for each activity only one instance that respects its different requirements and has the cheapest cost. For example, an activity a is the first activity (i.e, has the highest priority) in the process, needs an amount of RAM and vCPU capacities, has a $MinD_a$ and $MaxD_a$ and a penalty cost not null. This activity will take the less expensive resource R , without an interruption risk, that respects its required capacities and temporal constraints. We note that each instance

is assigned to only one activity

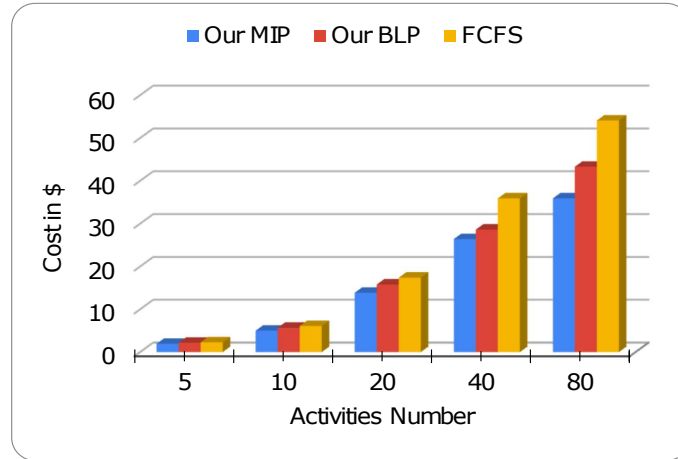


Figure 6.6: Approaches Comparison

It can be seen from the data in Figure. 6.6 that our BLP model provides the optimal BP deployment cost but less values are provided by our MIP model. This result may be explained by the fact that temporal flexibility helps to minimize the BP deployment cost. More specifically, matching activities' temporal constraints to cloud resources' temporal constraints allow to select cheaper cloud resources to perform activities. Otherwise, "Priority+FCFS" scheduling algorithm performs worse than our optimization methods. It tends to select less-expensive resources based on the arrival time of each activity while respecting its required capacities and temporal constraints. A possible explanation for this might be that the scheduling of BP's activities is done before the cloud resource allocation. However, using our MIP model we define simultaneously the scheduling plan and the selection of cloud resources and pricing strategies. In the end, we note that our MIP model saves up to 40% in-process cost in comparison with both methods.

6.4.4 Impact of the verification step

We note that linear programming optimization problems are NP- hard [143]. So they require high computational efforts to find out an optimal and even a feasible solution for large size problems. For that, it is often more important to reduce the search space to avoid waiting for a long time to obtain the optimal solution. To this end, to deal with more complex and large BPs, before moving to the optimization step, we check in our verification approach the temporal correctness of cloud resource allocation in time-aware BPs. Then, we take as inputs for our linear models (BLP and MIP) models: (i) a BP and (ii) only the set of cloud resources and pricing strategies that

ensure cloud resource allocations are temporally correct. In this manner, the size of the optimization problem is reduced and so the computation times of our linear models (BLP and MIP) are reduced. Consequently, our linear models may converge in a short time to an optimal solution [144]. For that, we vary the optimization problem size to compare between the values of computation time and objective function when the inputs of our linear models (BLP and MIP) are a BP, and (i) a set of cloud resources and their pricing strategies (BLP and MIP without verification), or (ii) a set of possible allocation options that are verified as per Section 5.4 (BLP and MIP with verification).

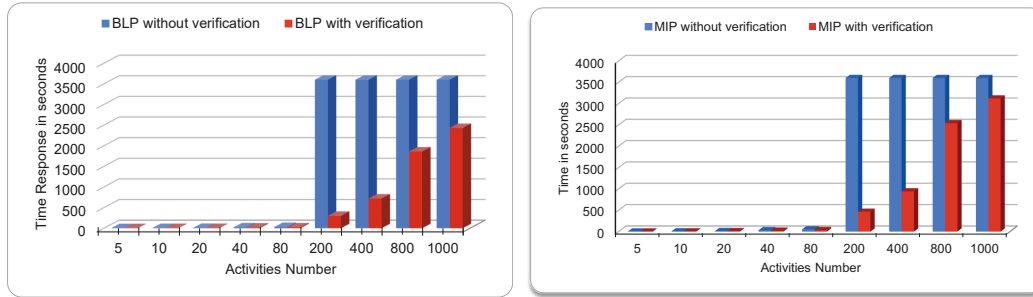
To this end, this Section is devoted to study the impact of the verification step. We evaluate, first, the impact of our model's inputs, and second the impact of correct allocations' number on the objective function and on the computation time values.

6.4.4.1 Impact of inputs

In Figure 6.7a and Figure 6.7b, we respectively present the computation time's and objective function's values of our BLP and MIP models. On the one hand, as expected, the results show that the computation time of both linear models has low values (about 30 seconds) in case of BPs with a small number of activities (under 200) compared to values (high number of hours) in case of BPs with a large number of activities. We also note that taking as input correct allocations, in both cases, helps to converge in a limited time (under 1 hour) to optimal solutions thanks to a restricted search space. More precisely, an organization can reduce up to 85% in computation time to obtain the optimal solution. This observation may support the hypothesis that minimizing the problem size can lead to minimizing the waiting time to reach an optimal solution.

On the other hand, the results revealed that the objective function values are always high if we take as input correct allocations (verification step). This is due to a restricted search space. But, if we take as input all cloud resources (without verification step) we notice that the objective function values are less thanks to the large variety of cloud resources proposed in various pricing strategies. Namely, an organization can save up to 8% in the BP deployment cost (i.e., objective function) when enlarging the search space (without verification step). In general, therefore, it seems that the restriction of the search space may cause the raise of the objective function value.

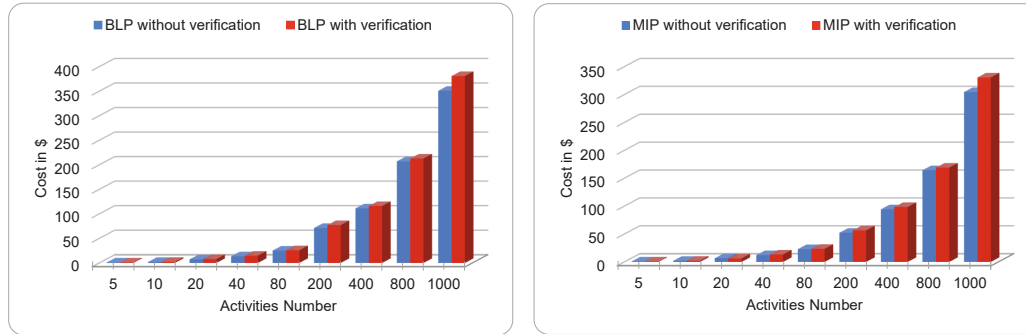
The results provided from our BLP and MIP models show the effectiveness of our solutions. In fact, taken together, these results suggest that our approach helps the designers to optimize the costs of BP deployment in cloud resources. Even in case of space restriction (with verification) for BPs with a large number of activities, the rate of increase of the objective function is quite small (up to 8%) compared to the rate of decrease of the computation time (up to 85%).



(a) BLP

(b) MIP

Figure 6.7: Impact of inputs on computation time



(a) BLP

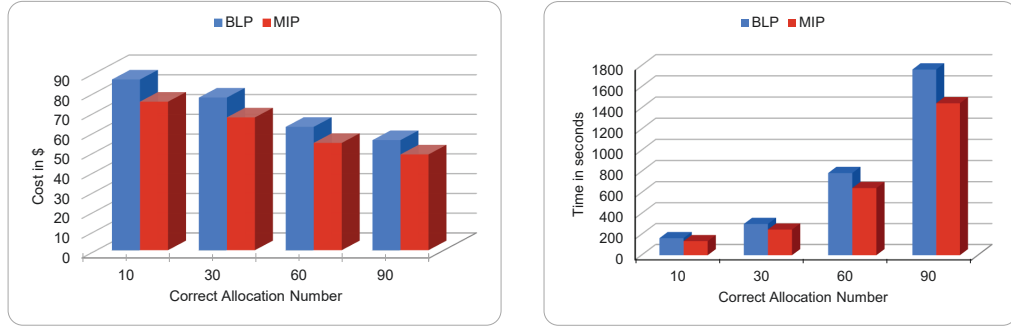
(b) MIP

Figure 6.8: Impact of inputs on objective function

6.4.4.2 Impact of correct allocations' number

The number of correct allocations ca is one of the factors that can impact both: the BP deployment cost and the computation time of our linear models. Thus, we take randomly a set of ca correct allocations. Then, we vary ca to analyze the values of the computation time and the objective function of our BLP and MIP. So, we take as inputs: (i) a BP of 200 activities and (ii) a set of correct allocations ca . The results are reported in Figure 6.9a and Figure 6.9b. As shown in Figure 6.9a, the BP deployment cost is always less expensive when ca is higher while the search space is larger and so there are more choices. This may be explained by the fact that each set of correct allocations may not provide the optimal BP deployment costs. Indeed, considering a large set of inputs raises the probability to encounter cheaper costs. In contrast, from

Figure 6.9b, it can be seen that following the increase of ca , a significant increase in the computation time was recorded. Mainly, the high value of ca helps to reduce the objective function but it raises the computation time of our linear models. It can therefore be assumed that the number of correct allocations has a significant impact on BP deployment cost and the computation time of our linear models.



(a) Objective function

(b) Computation time

Figure 6.9: Impact of correct allocations' number

6.4.5 CloudSim Results

In this section, we present tests and evaluations that we undertook to quantify the efficiency of CloudSim in modeling and simulating the cloud computing environment. To this end, we rely on CloudSim framework [89] to simulate a cloud environment. So, we take as inputs the UDM file describing: (i) the relationships between activities, (ii) their requirements (time, RAM, vCPU, etc), (iii) the selected cloud resources, and (iv) the pricing strategies that are selected to perform activities. Next, using the extended CloudSim, we simulate the BP execution in order to evaluate the difference between the cost values provided by our proposals and the cost computed using the simulator.

Figure. 6.10 illustrates the experimental results provided. It is apparent from those graphs that the results given by our BLP and MIP models are sometimes lesser and sometimes greater than the results given by the simulator. This difference is owing to costs values updated and extracted from Amazon API. We use Equations (6.18) and (6.19) to compute the difference in percentage between the cost values given by our linear models and the simulator. For instance, p_1 (respectively p_2) presents in percentage the difference between CloudSim value ob_{simul} and our BLP's (respectively our MIP's) ob_{BLP} (respectively ob_{MIP}) objective function value.

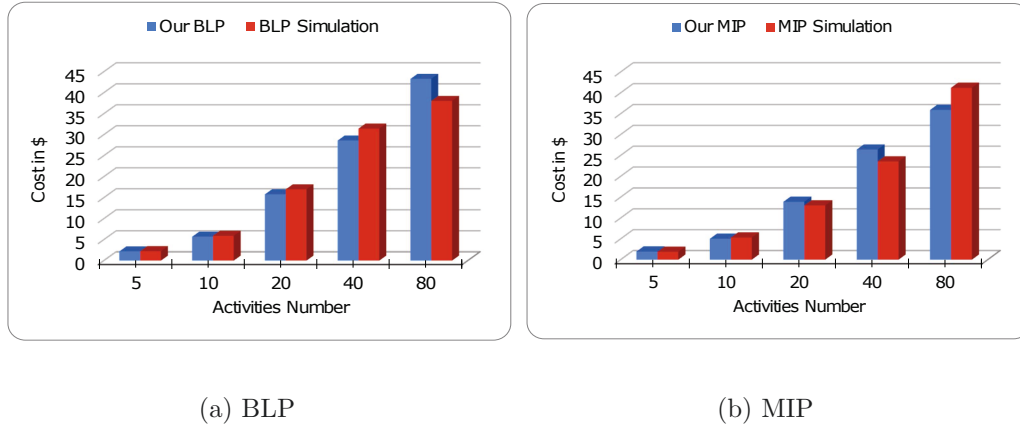


Figure 6.10: CloudSim Evaluation

$$p_1 = \left| 1 - \frac{ob_{BLP}}{ob_{simul}} \right| \times 100 \quad (6.18)$$

$$p_2 = \left| 1 - \frac{ob_{MIP}}{ob_{simul}} \right| \times 100 \quad (6.19)$$

Based on both equations (6.18) and (6.19), we observe in Figure 6.11 that, overall, p_1 and p_2 values are under 13%. These results are likely to be related to the cost updating as a consequence of, especially, the bidding process for spot instance. In contrast, the most obvious finding to emerge from the simulation is that our linear models give efficient results and its margin of error is quite small.

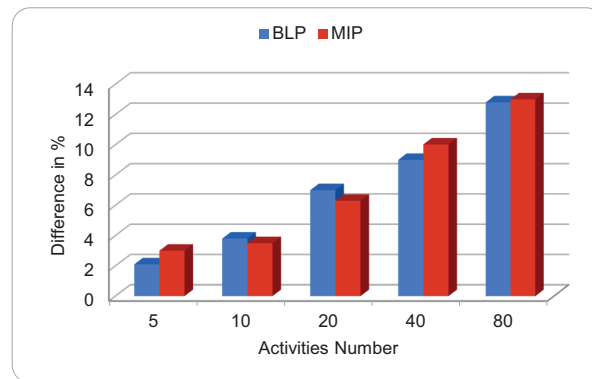


Figure 6.11: Difference in % between our linear programs and the simulator

6.5 Conclusion

In this chapter, we answered the questions mentioned in our thesis problematic (see Section 3.2) which is: *How to find the optimal deployment cost of a time-aware BPs in cloud resources?*

To find an optimal BP's deployment cost in cloud resources, we model our optimization problem using mathematical formulations. More precisely, we propose two linear models: a BLP and MIP that are composed of an objective function and a set of constraints. The objective function reduces the BP deployment cost under various constraints such as activities requirements, resource constraints, etc. Our models take as inputs a time-aware BP and a set of cloud resources proposed under different pricing strategies. If activities temporal constraints are flexible, we provide the best scheduling plan using our MIP model. Otherwise, the best assignment of cloud resources is provided using our BLP model. Moreover, we propose a CloudSim extension to enable the BP designers to analyze the cost for the best deployment scenario. The experimental results show the feasibility, effectiveness, and performance of our proposals.

Conclusion and Future Works



The research problem of our thesis has been expressed by this question: How to optimize the BP deployment cost in cloud resources based on temporal constraints? We presented in details, in previous chapters, our solutions to answer the raised question. In this Chapter, we first summary our work in Section 7.1 then we present the future work in Section 7.2.

7.1 Fulfillment of objectives

Considerable attention have been paid to PAISs in order to better manage and execute operational processes involving people, applications, and/or information sources on the basis of BP models [17]. An example of such systems is BPM. Its main instrument are BP models that represent BPs in terms of activities and their orders. Besides, the BP field is influenced by a wide range of temporal constraints such as durations, temporal dependency, etc. The satisfaction of those temporal constraints is fundamental to avoid critical situations and guarantees the safety of the involved parties. Further, organizations need to minimize their upfront investment on IT infrastructure. Therefore, they choose to outsource some operations to cloud computing due to its resource elasticity and pay-per-use benefits. Many cloud providers offer different pricing strategies (e.g., on-demand, reserved, and spot), defined based on temporal constraint, to accommodate users' changing and last-minute demands. In such environment, organizations need to support large amount of activities' temporal constraints and cloud resources having limited temporal availabilities proposed under various pricing strategies. Besides, they need to optimize the BP's deployment cost in cloud computing. Not only supporting the resource allocation in such BP models has been an interesting topic over the last years. But also optimization of the BP deployment cost in cloud resources constitutes an appealing challenge.

Many researches have been involved in both academics and industry. They differently address these challenges. On the one hand, several researches working on the resource perspective in BP models have been proposed. They suggest to support the resource perspective by extending process modeling languages. They provide formal definitions for the allocated resources. Nevertheless, such approaches only take into account objects and especially human resources and only few ones do consider the particular resource type that are cloud resources without considering their limited temporal availabilities. On the other hand, many researches working on optimization of the BP's deployment cost in cloud resources have been defined. They propose various approaches to minimize the BP's deployment cost in cloud computing. However, such proposals lack of considering advanced BP's temporal constraints as well as various pricing strategies to optimize BP's deployment cost.

To address these challenges, we proposed in this thesis an end-to-end approach, composed of three contributions, for supporting the design and optimization of cloud resource allocation in time-aware BP.

In the first contribution, we propose to integrate cloud resources' temporal constraints in BP. To this end, we define two steps to assist the BP's designer to model and graphically design his BP enriched with cloud resources and temporal aspects. To this end, we propose a formal model of a BP enriched with temporal constraints, cloud resources, as well as pricing strategies. Then, we propose a BPMN extension to support our proposed model. So, we offer for BP designer a tool to add cloud resources, pricing strategies, and BP constraints. We have developed a plug-in as a proof of concept to take into account our proposal to support cloud resources, pricing strategies, as well as activities' temporal constraints.

In the second contribution, we propose to formally verify at design time the temporal correctness of the cloud resource allocation in time-aware BP designed using our first contribution. Therefore, we define two steps to avoid the design errors such as temporal violation. For this reason, we define a set of rules to transform BPMN models into a network of timed automata models as a step towards a formal verification of the temporal correctness. Moreover, we implemented our transformation rules to automatically generate timed automata models used as inputs for the model checking tool to verify the satisfaction of some CTL properties such as liveness.

In the third contribution, we propose to find the optimal BP's deployment cost in cloud resources. Towards this end, we define two steps to assist the BP designer to find the resource assignment and/or the BP activities' scheduling that provides the optimal BP deployment cost and to simulate the best resource allocation to provide a more real BP deployment cost. More precisely, we model our optimization problem as two linear program models: a Binary Linear Program (BLP) and a Mixed Integer Program (MIP). The latter are composed of an objective function and a set of constraints. The objective function opts to reduce the BP's deployment cost under various constraints such as activities requirements, resource constraints, etc. Our models take as inputs a time-aware BP and a set of cloud resources proposed under different pricing strategies. If activities temporal constraints are flexible, we provide the best scheduling plan using our MIP that minimizes the BP deployment cost. Else, the best assignment of cloud resources is provided using our BLP that minimizes also the BP deployment cost. After the linear optimization step, we have proposed a simulation step through the extension of CloudSim [69] to enable the BP designer to analyze the cost for the best deployment scenario. As mentioned in Chapter 2, linear programming models have high complexity in case of a BP with a large number of activities. Therefore, to reduce the problem complexity, we resort to our verification step in order to verify the temporal correctness of a set of cloud resource allocations. Thus, we take as inputs a BP with a set of correct ones. Based on the results from our experimentation, the effectiveness, feasibility, and scalability are demonstrated.

7.2 Future Works

Our work clearly has some limitations. First, it only studied a subset of BPMN constructs. In particular, we did not consider BPMN's loop structure and data flow behaviours. For the sake of completeness, a natural extension to our approach would be to investigate these constructs in the proposed approach. Consequently, considering temporal data correlated with temporal constraints of activities and resource constraints is our next target.

Second, in our work, we focus only on computing resources exposed as part of the Infrastructure-as-a-Service (IaaS) model and additional networking and database costs are out of the scope of our work. So, we plan to improve the quantity of our work by enriching it with cloud resource types such as storage and network. This will increase the expressiveness of our approach. Moreover, we plan to extend our work by considering Fog and IoT resources. Indeed, this was the subject of an Erasmus+ mobility in the University of the Balearic Islands. During this internship, first, we studied existing works related to the pricing strategies of Fog and IoT resources. Then, we started to extend our proposals with Fog and IoT resources. Currently, we are evaluating our new approach.

Moreover, regarding our first contribution presented in Chapter 5, we aim shortly to verify the semantic correctness of the transformation rules: The generated timed automata should preserve semantic properties like termination (i.e., transformations should always lead to a result) and confluence (i.e., result should be unique). Besides, the scope of our approach is limited to verify the temporal correctness without detecting the errors. For overcoming this limitation, we see significant potential in following the temporal verification BP by an error detection approach, which presents a step toward achieving a conflict-free resource allocation in time-aware BP model. It would be interesting to also support error correction, following the error detection. As eventual error correction, we can notice the modification of the duration of some activities, the change of the overall BP structure, the substitution of some activities arriving at the modification of some cloud resources, or even their pricing strategies.

Regarding our second contribution presented in Chapter 6, at medium term, we aim at further evaluating our proposal by comparing it to recent approaches proposed in the area of our research such as [58, 286]. Besides, in our current work, we assume that a BP has only AND branching and a cloud resource can not be shareable. Therefore, we plan to consider other branching types: OR and XOR. Moreover, we assume that $MaxD_a=d_{a_q}$ and $D_{max}=D_{min}=du$. Thus, we aim to evaluate our proposals while considering different temporal durations and dependencies values. Further, we intend to deal with cloud resources shareability. Future work should also take into account data transfer cost. The latter are considered null in our current research since we assume that cloud resources are in the same availability zone. Moreover, until now we assumed that the spot price is not full dynamic and is known at the BP design time. Thus, we intend to study deeply the dynamic fluctuation of the spot

instance cost and to analyze the impact of this aspect on resource allocation and BP's deployment cost at runtime.

As research on cloud resources' management in BPM still at its beginning stage, we could follow other new perspectives. For instance, at long term, we aim at extending our approach to take into account the PaaS and SaaS resources. Further, we intend to deal with configurable BP models. More specifically, we are planning to propose an approach for supporting and optimizing configurable resource allocation in configurable BP models while considering the variety of pricing strategies.

Appendices

Implementation frameworks

Over the years, using frameworks is evolving. It becomes crucial in software and model engineering. In fact, it is helpful in developing softwares, tools and plug-ins better and faster. In our case, the use of frameworks is very beneficial because it provides a conceptual guide to utilize effective implementation options. In this section, we depict the frameworks involved in the plug-in's implementation. More precisely, our work is implemented as a plug-in for Eclipse and it allows designers to easily include the needed temporal constraints, pricing strategies, and resources in their BPs. The aim of this BPMN extension is to help BP designer to allocate cloud resources adequately and profitably. To achieve this goal, we extended the existing tool, BPMN2 Modeler [141]. In fact, the Eclipse BPMN2 Modeler is a graphical modeling tool for authoring BPs that allows creation and modification of BPMN diagrams. This tool is built on Eclipse Graphiti [100] and uses the BPMN 2.0 EMF metamodel which is developed within the Eclipse Model Development Tools project¹. Therefore, we will give an overview about EMF and the eclipse Graphiti.

A.1 Eclipse Modeling Framework: EMF

The most widely used MDE platform is EMF [100]. It is an open source project, a modeling framework for Eclipse and code generation facility for building tools and other applications based on a structured data model, integrated in Eclipse. We use EMF when we have structured data-model in Eclipse application which might get stored, displayed, and modified in a user interface. Since we need to extend the BPMN2 model, we will have to create our own EMF model. In fact, it boosts the Java programming productivity, application compatibility, and integration. The EMF is also needed in model-to-model transformations, and provides tools and runtime support to produce from a model specification described in XMI, a set of Java classes for the model. It provides also a set of adapter classes, that allow viewing and editing the model. Besides, models in EMF can be defined in different ways such as Java Interfaces, UML class diagrams, XML Schema, etc. It is indeed a stable standard for many model related technologies. Ecore is the core meta-model at the heart of EMF. It allows expressing meta-models by leveraging its constructs. These constructs,

¹<https://www.eclipse.org/graphiti/>

always prefixed by an “E” are *EPackage*, *EAttribute*, *EClass*, *EReference*, etc. EMF fundamental components on Eclipse are²:

- EMF: The core modeling framework including a meta-model Ecore for describing models and runtime support for the models.
- EMF.Edit: is a framework that includes generic reusable classes for building editors for EMF models. It provides content and label provider classes, property source support, and other convenience classes.
- EMF.Codegen: The EMF code generation facility is capable of generating everything needed to build a complete editor for an EMF model. It includes a Graphical User Interface from which generation options can be specified, and generators can be invoked. The generation facility leverages Java Development Tooling component of Eclipse. Three levels of code generation are supported:
 - Model: provides Java interfaces and implementation classes for all the classes in the model as well as a factory and package implementation class.
 - Adapters: generates implementation classes (called Item Providers) that adapt the model classes for editing and display.
 - Editor: produces a properly structured editor that conforms to the recommended style for Eclipse EMF model editors and serves as a starting point from which to start customizing.

The generator supports regeneration of code while preserving user modifications. As shown in Figure. A.1, model is created and defined in the Ecore format, which is basically a subset of UML Class diagrams. Then, from an Ecore model³, we can generate Java code.

A.2 BPMN2 Modeler palette extension using Graphiti

After extending the BPMN 2.0 meta-model using EMF, each meta-class is associated with a class that defines the corresponding graphical shape using Graphiti. In fact, Eclipse provides a modeling infrastructure evolving around EMF that offers graphical representations and editing possibilities.

More specifically, Graphiti is an Eclipse graphics framework that facilitates building graphical editors with little coding. It helps to describe the different graphical forms. The Graphiti project has main objectives such as providing plain Java Application Programming Interface for building graphical tools, and all the features expected to a graphical editor, including Drag-and-drop support, tool palette, etc. Besides, providing the required documentation and tutorials to develop a new editor

²<https://www.eclipse.org/>

³<https://mickael-baron.fr/modeling/>

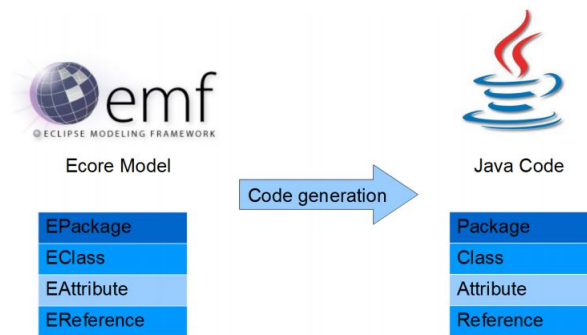


Figure A.1: Generating Java code from an Ecore model

simply because it is already implemented in the Framework. Also, helping the engineer to implement custom features such as non-standard behaviors while developing an Eclipse plug-in. So, a Java code is generated with the help of the EMF framework in order to finalize the creation of the plug-in.

Consequently, the BPMN2 Modeler palette is extended to include the required BPMN constructs for drawing the temporal constraints of the BP (i.e. Duration, Temporal Dependency, etc) and the activities that consume cloud resources. To easily use this palette, we classify them by categories such as *Task+CloudResource*, *TemporalDependency*, *Duration* and *Absolute Temporal constraints* as shown in Figure A.2.

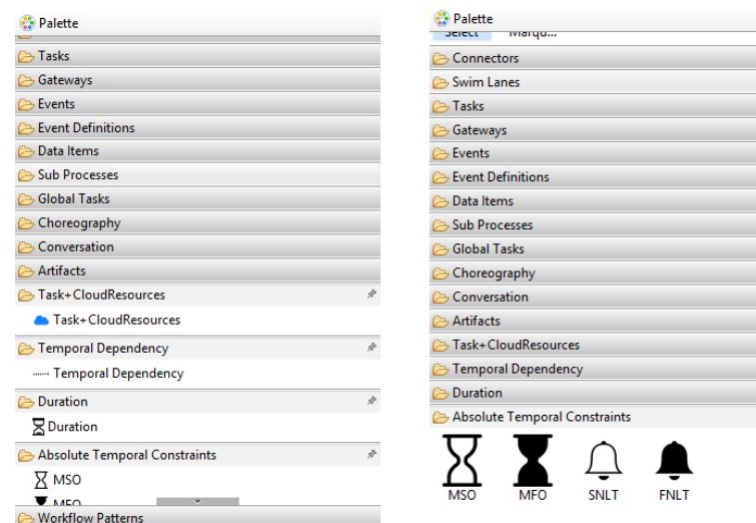


Figure A.2: The BPMN2 Modeler extended palette

Plug-in creation

Based on the BPMN 2.0 meta-model extension, we present our Eclipse plug-in. In fact, it contains the definition of pricing strategies, cloud resources as well as activities' temporal constraints. The latter will be added to the BPMN2 modeler editor and expressed in BPMN meta-model with EMF. Indeed, for each meta-class in the meta-model, a Java class is generated and associated to it. These Java classes define its graphical representation via Graphiti. The plug-in is described in a XML file, called *plugin.xml*. To develop this plug-in, we essentially need to create the meta-model to define the extension elements. Then, there exist several other steps that we will depict as we go along.

B.1 Meta-model creation

Meta-models can be specified in different ways: XML Schema Definition (XSD), UML, Ecore models etc. In our work, we use Ecore as shown in Figure B.1. Since we need to extend the BPMN 2.0 meta-model, we start by creating the new EMF model file (*MyModel.ecore*). We define the namespace *URI* that identifies the model and the target runtime. In fact, the extension model requires a *DocumentRoot* class. It presents the root element referencing all other classes through a composition link. In addition, EMF offers the functionality of loading the BPMN meta-model as shown in Figure B.2. Thus, we can use the needed BPMN predefined meta-classes as super classes of our extension's classes (depicted in Figure B.1).

B.2 Java code generation

EMF boosts the productivity and saves time since it offers an automatic code generation. In fact, the model have to be translated into Java code dedicated to create instances of the model that we specified with ecore. The code generation requires the creation of an EMF generator model called *genmodel*. This model is also an EMF model. It contains only information for generation and that could not be integrated in the model such as the Path generation, package, prefix. Therefore, we create the EMF generator model *MyModel.genmodel* from *MyModel.ecore*. Henceforth, we can

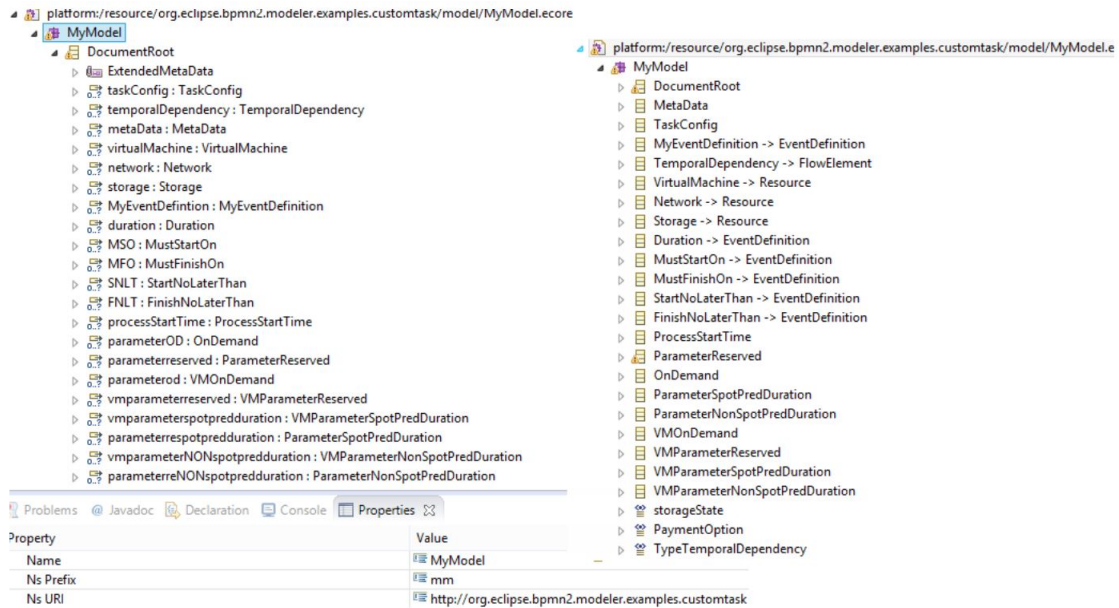


Figure B.1: Ecore model and the DocumentRoot class

generate Java code since *.genmodel* file contains additional information for the code generation as well as the control parameter about how the code should be generated.

As shown in Figure B.3, we also need to reference BPMN 2.0 models. Thus, we right-click on the root node of the *MyModel.genmodel* file and select Generate Model Code. This creates the Java implementation of the EMF model in the current project. An Ecore class (i.e. an Eclass) actually corresponds to two things in Java: an interface and a corresponding implementation class. Doing the step above, EMF generates four different packages.

1. The first one has the name of the plug-in. It contains the Activator class which is loaded when the plug-in is activated. Also, it contains the Graphiti classes of the extension's elements.
2. The second one contains the Interfaces of meta-classes.
3. The third one (with *.impl* suffix) contains concrete implementation of the interfaces defined in the second package.
4. The fourth one (with *.util* suffix) contains the AdapterFactory and other utility classes.

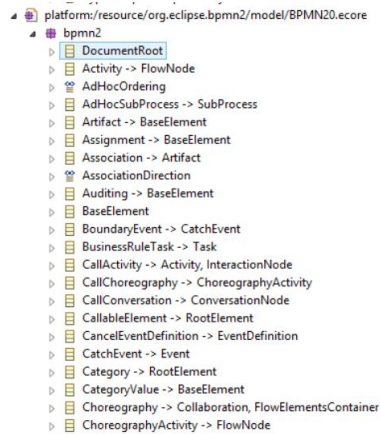


Figure B.2: The loaded BPMN meta-model

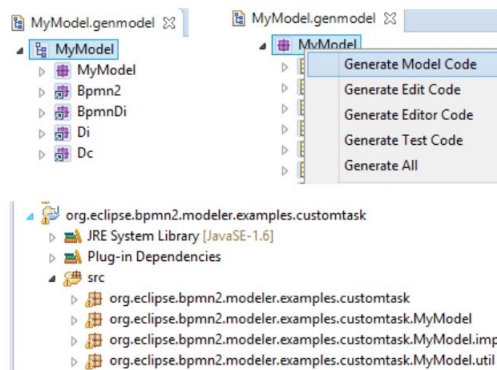


Figure B.3: Genmodel file and the generated packages

B.3 The dependencies of the extension plug-in

The plug-in extension needs several dependencies as shown in Figure B.4. In fact, they are a list of plug-ins that are required for the operation of our plug-in. They are defined in the plug-in *manifest* file *plugin.xml* that describes how the plug-in extends the platform, what extensions it publishes itself, and how it implements its functionality. It also helps the Eclipse runtime to activate the plug-in.

Each extension plug-in needs a runtime element for its identification and uniqueness. So, our plug-in will not be confusing with other plug-ins. Indeed, the BPMN2 Runtime Extension is the core element of a BPMN2 Plug-in extension. Thus, we create a new class *MyRuntimeExtension* that implements *IBpmn2RuntimeExtension* predefined Interface (from *org.eclipse.bpmn2.modeler.core dependency*). This class is

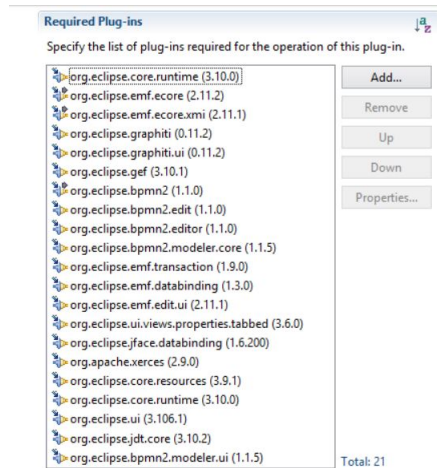


Figure B.4: Dependencies required for the plug-in extension

used by the editor to identify BPMN resources that our extension can handle.

We create the runtime element then we relate it to the *MyRuntimeExtension* class. It is the target runtime extension point to define our *plugin.xml*. We clarify this point in Figure B.5 by showing runtime extension element details as well as an excerpt from the *plugin.xml* file.

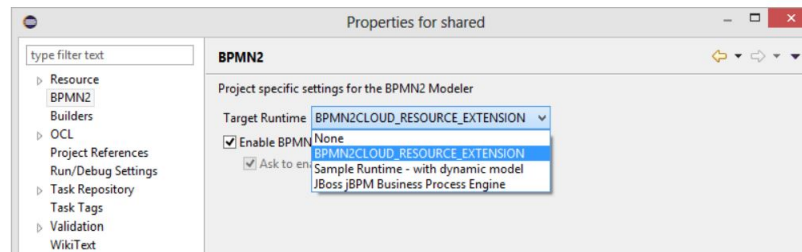
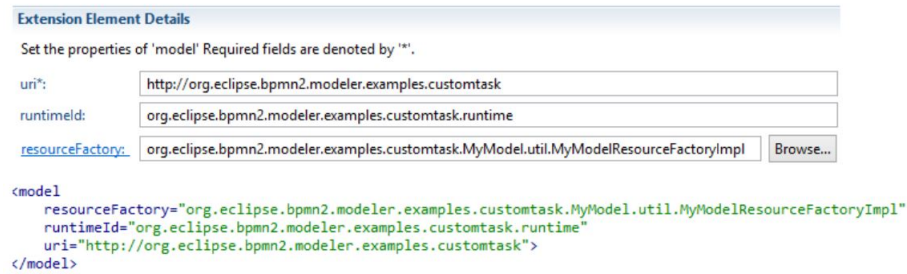


Figure B.5: The target runtime element

We can test our plug-in as an Eclipse application from the *plugin.xml* Editor. We create a new empty project in the runtime workspace and open the project properties dialog (right-click on the project name and select Properties from the context menu). We should see the section *BPMN2* with a *Target Runtime selector*. Our new Runtime Extension *BPMN2CLOUD_RESOURCE_EXTENSION* should be listed there as shown in Figure B.5.

In addition, we have to define an extension model element which makes our extension model known to the editor. First, it has to match the namespace *URI* we defined

in the Ecore file. Second, it must describe the corresponding ResourceFactory *MyModelResourceFactoryImpl.java* class that extends *BPMN2ModelerResourceFactoryImpl* which comes from the *org.eclipse.bpmn2.modeler.core* dependency as shown in Figure B.6.



Extension Element Details

Set the properties of 'model' Required fields are denoted by '*':

uri*:

runtimeId:

resourceFactory:

```
<model
  resourceFactory="org.eclipse.bpmn2.modeler.examples.customtask.MyModel.util.MyModelResourceFactoryImpl"
  runtimeId="org.eclipse.bpmn2.modeler.examples.customtask.runtime"
  uri="http://org.eclipse.bpmn2.modeler.examples.customtask">
</model>
```

Figure B.6: The model element

Bibliography

- [1] Weske Mathias. Business process management: Concepts, languages, architectures 2nd ed., xv, 403 p. 300 illus. hardcover isbn 978-3-642-28615-5mcleod, raymond and schell, george p. 2008 management information systems, upper saddle river new jersey 07458, 2012.
- [2] Ayodeji Adesina and Derek Molloy. A business process management based virtual learning environment-customised learning paths. In *CSEU (1)*, pages 365–368, 2011.
- [3] Peter Mell and Tim Grance. The nist definition of cloud computing. Technical report, National Institute of Standards and Technology, 2011.
- [4] Amazon ec2. <https://aws.amazon.com/ec2/>, (Mai 20, 2017).
- [5] MOHIT KUMAR, Kalka Dubey, and s Sharma. *Job Scheduling Algorithm in Cloud Environment Considering the Priority and Cost of Job*, pages 313–320. 04 2017.
- [6] Frédéric Jouault and Ivan Kurtev. Transforming models with atl. In *International Conference on Model Driven Engineering Languages and Systems*, pages 128–138. Springer, 2005.
- [7] Andreas Lanz, Barbara Weber, and Manfred Reichert. Time patterns for process-aware information systems. *Requirements Engineering*, 19(2):113–141, 2014.
- [8] 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 *International Conference on Advanced Information Systems Engineering*, pages 53–68, 2015.
- [9] Wasim Sadiq and Maria E Orlowska. Analyzing process models using graph reduction techniques. *Information systems*, 25(2):117–134, 2000.
- [10] Saoussen Cheikhrouhou. *Specification and Verification of Temporal Constraints in Inter-Organisational Business Processes*. PhD thesis, University of Sfax, Tunisia, 2014.
- [11] Mariska Netjes, Wil MP van der Aalst, and Hajo A Reijers. Analysis of resource-constrained processes with colored petri nets. In *Sixth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, volume 576, pages 251–266. series DAIMI, 2005.

-
- [12] Seung Yeob Shin, Yuriy Brun, and Leon J Osterweil. Specification and analysis of human-intensive system resource-utilization policies. In *2016 IEEE/ACM International Workshop on Software Engineering in Healthcare Systems (SEHS)*, pages 8–14. IEEE, 2016.
- [13] Abir Ismaili-Alaoui, Khalid Benali, Karim Baïna, and Jamal Baïna. Business process instances scheduling with human resources based on event priority determination. In *International Conference on Big Data, Cloud and Applications*, pages 118–130. Springer, 2018.
- [14] Giray Havur, Cristina Cabanillas, Jan Mendling, and Axel Polleres. Automated resource allocation in business processes with answer set programming. In *International Conference on Business Process Management*, pages 191–203. Springer, 2016.
- [15] Suraj Pandey, Linlin Wu, Siddeswara Mayura Guru, and Rajkumar Buyya. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In *2010 24th IEEE international conference on advanced information networking and applications*, pages 400–407. IEEE, 2010.
- [16] Souha Boubaker. *Formal verification of business process configuration in the Cloud. (Vérification formelle de la configuration des processus métiers dans le Cloud)*. PhD thesis, University of Paris-Saclay, France, 2018.
- [17] Hongyan Ma. Process-aware information systems: Bridging people and software through process technology. *Journal of the American Society for Information Science and Technology*, 58(3):455–456, 2007.
- [18] Diimitrios Georgakopoulos, Mark Hornick, and Amit Sheth. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and parallel Databases*, 3(2):119–153, 1995.
- [19] Howard Smith and Peter Fingar. *Business process management: the third wave*, volume 1. Meghan-Kiffer Press Tampa, FL, 2003.
- [20] Wil Van Der Aalst, Kees Max Van Hee, and Kees van Hee. *Workflow management: models, methods, and systems*. MIT press, 2004.
- [21] Omg. business process model and notation (bpmn) 2.0. <http://www.omg.org/spec/BPMN/2.0/>.
- [22] August-Wilhelm Scheer. *ARIS—vom Geschäftsprozess zum Anwendungssystem*. Springer-Verlag, 2013.
- [23] Wil MP Van Der Aalst and Arthur HM Ter Hofstede. Yawl: yet another workflow language. *Information systems*, 30(4):245–275, 2005.

-
- [24] v2.3 omg. unified modelling language. <https://www.omg.org/spec/UML/2.3>.
- [25] Daniel J Clancy and Benjamin J Kuipers. Qualitative simulation as a temporally-extended constraint satisfaction problem. In *AAAI/IAAI*, pages 240–247, 1998.
- [26] Bilal Kanso and Safouan Taha. Temporal constraint support for ocl. In *International Conference on Software Language Engineering*, pages 83–103. Springer, 2012.
- [27] Souha Boubaker, Walid Gaaloul, Mohamed Graiet, and Nejib Ben Hadj-Alouane. Event-b based approach for verifying cloud resource allocation in business process. In *Proceedings of the IEEE International Conference on Services Computing*, pages 538–545, New York City, NY, USA, 2015.
- [28] Wang Long, Lan Yuqing, and Xia Qingxin. Using cloudsims to model and simulate cloud computing environment. In *2013 Ninth International Conference on Computational Intelligence and Security*, pages 323–328. IEEE, 2013.
- [29] Denis Gagné and André Trudel. Time-bpmn. In *Commerce and Enterprise Computing, 2009. CEC'09. IEEE Conference on*, pages 361–367. IEEE, 2009.
- [30] Saoussen Cheikhrouhou, Slim Kallel, Nawal Guermouche, and Mohamed Jmaiel. Enhancing formal specification and verification of temporal constraints in business processes. In *Services Computing (SCC)*, 2014.
- [31] Saoussen Cheikhrouhou, Slim Kallel, Nawal Guermouche, and Mohamed Jmaiel. Toward a time-centric modeling of business processes in BPMN 2.0. In *The 15th International Conference on Information Integration and Web-based Applications & Services*, page 154, 2013.
- [32] S. Boubaker, A. Mammari, M. Graiet, and W. Gaaloul. Formal verification of cloud resource allocation in business processes using event-b. In *2016 IEEE 30th International Conference AINA*, pages 746–753, 2016.
- [33] Emna Hachicha and Walid Gaaloul. Towards resource-aware business process development in the cloud. In *Advanced Information Networking and Applications (AINA), 2015 IEEE*, pages 761–768, 2015.
- [34] Ikram Garfatta, Kais Klai, Mohamed Graiet, and Walid Gaaloul. Formal modelling and verification of cloud resource allocation in business processes. In *Proceedings of the Confederated International Conferences On the Move to Meaningful Internet Systems*, pages 552–567. Springer, 2018.
- [35] Andreas Lanz, Jens Kolb, and Manfred Reichert. Enabling personalized process schedules with time-aware process views. In *Advanced Information Systems Engineering Workshops*, pages 205–216. Springer, 2013.

- [36] Fairouz Fakhfakh, Hatem Hadj Kacem, Ahmed Hadj Kacem, and Faten Fakhfakh. Preserving the correctness of dynamic workflows within a cloud environment. *Procedia Computer Science*, 126:1541–1550, 2018.
- [37] Zhengxing Huang, Wil MP van der Aalst, Xudong Lu, and Huilong Duan. Reinforcement learning based resource allocation in business process management. *Data & Knowledge Engineering*, 70(1):127–145, 2011.
- [38] Cristina Cabanillas. Process-and resource-aware information systems. In *2016 IEEE 20th International Enterprise Distributed Object Computing Conference (EDOC)*, pages 1–10. IEEE, 2016.
- [39] Cristina Cabanillas, Alex Norta, Manuel Resinas, Jan Mendling, and Antonio Ruiz-Cortés. Towards process-aware cross-organizational human resource management. In *Enterprise, Business-Process and Information Systems Modeling*, pages 79–93. Springer, 2014.
- [40] Luis Jesús Ramón Stroppi, Omar Chiotti, and Pablo David Villarreal. A bpmn 2.0 extension to define the resource perspective of business process models. In *XIV Congreso Iberoamericano en Software Engineering*, 2011.
- [41] Nick Russell, Wil MP van der Aalst, Arthur HM Ter Hofstede, and David Edmond. Workflow resource patterns: Identification, representation and tool support. In *International Conference on Advanced Information Systems Engineering*, pages 216–232. Springer, 2005.
- [42] Nick Russell and Wil MP van der Aalst. Evaluation of the bpel4people and ws-humantask extensions to ws-bpel 2.0 using the workflow resource patterns. *Bpm center report, Department of Technology Management, Eindhoven University of Technology GPO Box*, 513:142, 2007.
- [43] Iman Firouzian, Morteza Zahedi, and Hamid Hassanpour. Cycle time optimization of processes using an entropy-based learning for task allocation. *International Journal of Engineering*, 32(8):1090–1100, 2019.
- [44] Hedong Yang, Chaokun Wang, Yingbo Liu, and Jianmin Wang. An optimal approach for workflow staff assignment based on hidden markov models. In *OTM Confederated International Conferences” On the Move to Meaningful Internet Systems”*, pages 24–26. Springer, 2008.
- [45] Molka Rekik, Khoulood Boukadi, Nour Assy, Walid Gaaloul, and Hanène Ben-Abdallah. A linear program for optimal configurable business processes deployment into cloud federation. In *Services Computing (SCC), 2016 IEEE International Conference on*, pages 34–41. IEEE, 2016.

-
- [46] Guillaume Rosinosky, Samir Youcef, and Francois Charoy. A genetic algorithm for cost-aware business processes execution in the cloud. In *International Conference on Service-Oriented Computing*, pages 198–212. Springer, 2018.
- [47] Fatemeh Ebadifard and Seyed Morteza Babamir. Optimizing multi objective based workflow scheduling in cloud computing using black hole algorithm. In *2017 3th International Conference on Web Research (ICWR)*, pages 102–108. IEEE, 2017.
- [48] Fairouz Fakhfakh, Hatem Hadj Kacem, and Ahmed Hadj Kacem. A provisioning approach of cloud resources for dynamic workflows. In *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on*, pages 469–476. IEEE, 2015.
- [49] Elio Goettelmann, Walid Fdhila, and Claude Godart. Partitioning and cloud deployment of composite web services under security constraints. In *Cloud Engineering (IC2E), 2013 IEEE International Conference on*, pages 193–200. IEEE, 2013.
- [50] Jiajie Xu, Chengfei Liu, Xiaohui Zhao, and Zhiming Ding. Incorporating structural improvement into resource allocation for business process execution planning. *Concurrency and Computation: Practice and Experience*, 25(3):427–442, 2013.
- [51] Yuekun Chen, Guoqi Xie, and Renfa Li. Reducing energy consumption with cost budget using available budget preassignment in heterogeneous cloud computing systems. *IEEE Access*, 6:20572–20583, 2018.
- [52] WenAn Tan, Yong Sun, Ling Xia Li, GuangZhen Lu, and Tong Wang. A trust service-oriented scheduling model for workflow applications in cloud computing. *IEEE Systems Journal*, 8(3):868–878, 2013.
- [53] Juan J Durillo and Radu Prodan. Multi-objective workflow scheduling in amazon ec2. *Cluster computing*, 17(2):169–189, 2014.
- [54] Sen Su, Jian Li, Qingjia Huang, Xiao Huang, Kai Shuang, and Jie Wang. Cost-efficient task scheduling for executing large programs in the cloud. *Parallel Computing*, 39(4-5):177–188, 2013.
- [55] Lingfang Zeng, Bharadwaj Veeravalli, and Xiaorong Li. Scalestar: Budget conscious scheduling precedence-constrained many-task workflow applications in cloud. In *2012 IEEE 26th International Conference on Advanced Information Networking and Applications*, pages 534–541. IEEE, 2012.
- [56] Alexander A Visheratin, Mikhail Melnik, and Denis Nasonov. Workflow scheduling algorithms for hard-deadline constrained cloud environments. *Procedia Computer Science*, 80:2098–2106, 2016.

- [57] Gursleen Kaur and Mala Kalra. Deadline constrained scheduling of scientific workflows on cloud using hybrid genetic algorithm. In *2017 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence*, pages 276–280. IEEE, 2017.
- [58] Junlong Zhou, Tian Wang, Peijin Cong, Pingping Lu, Tongquan Wei, and Mingsong Chen. Cost and makespan-aware workflow scheduling in hybrid clouds. *Journal of Systems Architecture*, page 101631, 2019.
- [59] Yaser Jararweh, Zakarea Alshara, Moath Jarrah, Mazen Kharbutli, and Mohammad Noraden Alsaleh. Teachcloud: a cloud computing educational toolkit. *International Journal of Cloud Computing*, 2(2/3):237–257, 2013.
- [60] Juergen Mangler and Stefanie Rinderle-Ma. CPEE - cloud process execution engine. In *Proceedings of the BPM Demo Sessions, co-located with the 12th International Conference on Business Process Management*, page 51, 2014.
- [61] Yaser Jararweh, Moath Jarrah, Mazen Kharbutli, Zakarea Alshara, Mohammed Noraden Alsaleh, and Mahmoud Al-Ayyoub. Cloudexp: A comprehensive cloud computing experimental framework. *Simulation Modelling Practice and Theory*, 49:180–192, 2014.
- [62] Rania Ben Halima, Slim Kallel, Kais Klai, Walid Gaaloul, and Mohamed Jmaiel. Formal verification of time-aware cloud resource allocation in business process. In *OTM Confederated International Conferences” On the Move to Meaningful Internet Systems”*, pages 400–417, 2016.
- [63] Rania Ben Halima, Imen Zouaghi, Slim Kallel, Walid Gaaloul, and Mohamed Jmaiel. Formal verification of temporal constraints and allocated cloud resources in business processes. In *Advanced Information Networking and Applications (AINA), 2018 IEEE 32th International Conference on*, 2018 (to appear).
- [64] Rania Ben Halima, Slim Kallel, Walid Gaaloul, Zakaria Maamar, and Mohamed Jmaiel. Toward a correct and optimal time-aware cloud resource allocation to business processes. *Future Generation Computer Systems*, 2019 (submitted).
- [65] Rania Ben Halima, Slim Kallel, Walid Gaaloul, and Mohamed Jmaiel. Scheduling business process activities for time-aware cloud resource allocation. In *On the Move to Meaningful Internet Systems. OTM 2018 Conferences - Confederated International Conferences: CoopIS, C&TC, and ODBASE 2018, Valletta, Malta, October 22-26, 2018, Proceedings, Part I*, pages 445–462, 2018.
- [66] Rania Ben Halima, Slim Kallel, Mehdi Ahmed Nacer, and Walid Gaaloul. Optimal business process deployment cost in cloud resources. *Journal of Supercomputing (2020)*., 2020.

-
- [67] Rania Ben Halima, Slim Kallel, Walid Gaaloul, and Mohamed Jmaiel. Optimal cost for time-aware cloud resource allocation in business process. In *2017 IEEE International Conference on Services Computing, SCC 2017, Honolulu, HI, USA, June 25-30, 2017*, pages 314–321, 2017.
- [68] Daniel Pierre Bovet, Pierluigi Crescenzi, and D Bovet. *Introduction to the Theory of Complexity*. Prentice Hall London, 1994.
- [69] Rodrigo N Calheiros, Rajiv Ranjan, César AF De Rose, and Rajkumar Buyya. Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services. *arXiv preprint arXiv:0903.2525*, 2009.
- [70] Hamed Mili, Guy Tremblay, Guitta Bou Jaoude, Éric Lefebvre, Lamia Elabed, and Ghizlane El Boussaidi. Business process modeling languages: Sorting through the alphabet soup. *ACM Computing Surveys (CSUR)*, 43(1):4, 2010.
- [71] Jan Recker. Opportunities and constraints: the current struggle with bpmn. *Business Process Management Journal*, 16(1):181–201, 2010.
- [72] Henrik Bohnenkamp and Axel Belinfante. Timed testing with torx. In *International Symposium on Formal Methods*, pages 173–188. Springer, 2005.
- [73] Saoussen Cheikhrouhou, Slim Kallel, and Mohamed Jmaiel. Toward a verification of time-centric business process models. In *2014 IEEE 23rd International WETICE*, pages 326–331, 2014.
- [74] Monika Weidmann, Falko Kötter, Maximilien Kintz, Daniel Schleicher, Ralph Mietzner, and Frank Leymann. Adaptive business process modeling in the internet of services (abis). In *Proceedings of the Sixth International Conference on Internet and Web Applications and Services (ICIW)*, 2011.
- [75] Wil MP van Der Aalst. Workflow patterns. *Encyclopedia of Database Systems*, pages 3557–3558, 2009.
- [76] Boudewijn F van Dongen, Wil MP Van der Aalst, and Henricus MW Verbeek. Verification of eps: Using reduction rules and petri nets. In *International Conference on Advanced Information Systems Engineering*, pages 372–386. Springer, 2005.
- [77] Anton JMM Weijters and Wil MP Van der Aalst. Rediscovering workflow models from event-based data using little thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003.
- [78] Rania Ben Halima, Imen Zouaghi, Slim Kallel, Walid Gaaloul, and Mohamed Jmaiel. Formal verification of temporal constraints and allocated cloud resources in business processes. In *32nd IEEE International Conference on Advanced*

- Information Networking and Applications, AINA 2018, Krakow, Poland, May 16-18, 2018*, pages 952–959, 2018.
- [79] Ludmila Penicina. Towards the mapping of multidimensional bpmn models to process definition standards. *Scientific Journal of Riga Technical University. Computer Sciences*, 41(1):76–83, 2010.
- [80] Emna Hachicha Belghith. *Supporting cloud resource allocation in configurable business process models. (Supporter l’allocation des ressources cloud dans les processus métiers configurables)*. PhD thesis, University of Paris-Saclay, France, 2017.
- [81] Michael Zur Muehlen and Jan Recker. How much language is enough? theoretical and practical use of the business process modeling notation. In *Seminal Contributions to Information Systems Engineering*, pages 429–443. Springer, 2013.
- [82] Nour Assy. *Automated support of the variability in configurable process models. (Automatiser le support de la variabilité dans les modèles de processus configurables)*. PhD thesis, University of Paris-Saclay, France, 2015.
- [83] Azzam Sleit, Nada Misk, Fatima Badwan, and Tawfiq Khalil. Cloud computing challenges with emphasis on amazon ec2 and windows azure. *International Journal of Computer Networks & Communications*, 5(5):35, 2013.
- [84] Adel Nadjaran Toosi. *On the Economics of Infrastructure as a Service Cloud Providers: Pricing, Markets and Profit Maximization*. PhD thesis, University of Melbourne, Department of Computing and Information Systems, 2014.
- [85] Adel Nadjaran Toosi. *On the Economics of Infrastructure as a Service Cloud Providers: Pricing, Markets, and Profit Maximization*. PhD thesis, UNIVERSITY OF MELBOURNE, Australia, 2014.
- [86] Online. Google cloud. <https://cloud.google.com/>.
- [87] Online. Microsoft azure. <https://azure.microsoft.com/en-us/>.
- [88] Seung-Hwan Lim, Bikash Sharma, Gunwoo Nam, Eun Kyoung Kim, and Chita R Das. Mdcsim: A multi-tier data center simulation, platform. In *2009 IEEE International Conference on Cluster Computing and Workshops*, pages 1–9. IEEE, 2009.
- [89] Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, 41(1):23–50, 2011.

-
- [90] Baptiste Louis, Karan Mitra, Saguna Saguna, and Christer Åhlund. Cloudsimdisk: Energy-aware storage simulation in cloudsim. In *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, pages 11–15. IEEE, 2015.
- [91] Soumya Ray and Ajanta De Sarkar. Execution analysis of load balancing algorithms in cloud computing environment. *International Journal on Cloud Computing: Services and Architecture (IJCCSA)*, 2(5):1–13, 2012.
- [92] Harmanbir Singh Sidhu et al. Comparative analysis of scheduling algorithms of cloudsim in cloud computing. *International Journal of Computer Applications*, 975:8887, 2014.
- [93] Pinal Salot. A survey of various scheduling algorithm in cloud computing environment. *International Journal of Research in Engineering and Technology*, 2(2):131–135, 2013.
- [94] Robert France and Bernhard Rumpe. Model-driven development of complex software: A research roadmap. In *2007 Future of Software Engineering*, pages 37–54. IEEE Computer Society, 2007.
- [95] Stuart Kent. Model driven engineering. In *International Conference on Integrated Formal Methods*, pages 286–298. Springer, 2002.
- [96] Douglas C Schmidt. Model-driven engineering. *COMPUTER-IEEE COMPUTER SOCIETY-*, 39(2):25, 2006.
- [97] Freddy Allilaire, Jean Bézivin, Frédéric Jouault, and Ivan Kurtev. Atl-eclipse support for model transformation. In *Proceedings of the Eclipse Technology eXchange workshop (eTX) at the ECOOP 2006 Conference, Nantes, France*, volume 66. Citeseer, 2006.
- [98] Frédéric Jouault and Ivan Kurtev. Transforming models with atl. In *International Conference on Model Driven Engineering Languages and Systems*, pages 128–138. Springer, 2005.
- [99] OMG. Meta object facility (mof) 2.0 query/view/ transformation specification. Technical report, Object Management Group, Inc., 2011.
- [100] Kolovos Dimitris, Rose Louis, García-Domínguez Antonio, and Paige Richard. *The epsilon book*. Eclipse, 2018.
- [101] OMG. Object constraint language version 2.4. Technical report, Object Management Group, Inc., 2014.

-
- [102] Gerd Behrmann, Alexandre David, and Kim G Larsen. A tutorial on uppaal. In *Formal methods for the design of real-time systems*, pages 200–236. Springer, 2004.
- [103] Online. UPPAAL. <http://www.uppaal.org/>, June 02, 2015.
- [104] Bernard Berthomieu and Francois Vernadat. Time petri nets analysis with tina. In *QEST*, volume 6, pages 123–124, 2006.
- [105] Michael Leuschel and Michael Butler. Prob: A model checker for b. In *International Symposium of Formal Methods Europe*, pages 855–874. Springer, 2003.
- [106] Rajeev Alur and David L Dill. A theory of timed automata. *Theoretical computer science*, 126(2):183–235, 1994.
- [107] Carl Adam Petri. Communication with automata. Technical report, Rome Air Development Center, 1966.
- [108] Jos CM Baeten. A brief history of process algebra. *Theoretical Computer Science*, 335(2-3):131–146, 2005.
- [109] Jean-Raymond Abrial. *Modeling in Event-B: system and software engineering*. Cambridge University Press, 2010.
- [110] Gerd Behrmann, Alexandre David, and Kim Larsen. A tutorial on uppaal. *Formal methods for the design of real-time systems*, pages 33–35, 2004.
- [111] Marius Bozga, Conrado Daws, Oded Maler, Alfredo Olivero, Stavros Tripakis, and Sergio Yovine. Kronos: A model-checking tool for real-time systems. In *International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 298–302. Springer, 1998.
- [112] Guillaume Hutzler, Hanna Klaudel, and Dong Yue Wang. Towards timed automata and multi-agent systems. In *International Workshop on Formal Approaches to Agent-Based Systems*, pages 161–172. Springer, 2004.
- [113] Timed automata. <https://www.win.tue.nl/>, (Mai 06, 2019).
- [114] Thomas A Henzinger, Xavier Nicollin, Joseph Sifakis, and Sergio Yovine. Symbolic model checking for real-time systems. *Information and computation*, 111(2):193–244, 1994.
- [115] Edmund M Clarke and E Allen Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Workshop on Logic of Programs*, pages 52–71. Springer, 1981.

-
- [116] Leslie Lamport. Proving the correctness of multiprocess programs. *IEEE transactions on software engineering*, pages 125–143, 1977.
- [117] Wan Fokkink, Allard Kakebeen, and Jun Pang. Adapting the uppaal model of a distributed lift system. In *International Conference on Fundamentals of Software Engineering*, pages 81–97. Springer, 2007.
- [118] Davoud Mougouei. A mathematical programming approach to considering value dependencies in software requirement selection, 2018.
- [119] Kamil Figiela. *Optimization of Resource Allocation on the Cloud*. PhD thesis, AGH University of Science and Technology, Poland, 2013.
- [120] Singiresu S. Rao. *Engineering Optimization: Theory and Practice*. John Wiley and Sons, 2009.
- [121] Leila Hadded. *Optimization of autonomic resources for the management of service-based business processes in the Cloud. (Optimisation des ressources autonomiques pour la gestion des processus métier à base de services dans le Cloud)*. PhD thesis, University of Paris-Saclay, France, 2018.
- [122] AH Land and AG Doig. An automatic method of solving discrete programming problems. *econometrica*. v28. 1960.
- [123] Sanjoy Dasgupta, Christos H Papadimitriou, and Umesh Virkumar Vazirani. *Algorithms*. McGraw-Hill Higher Education New York, 2008.
- [124] Egon Balas, Sebastian Ceria, Gérard Cornuéjols, and N Natraj. Gomory cuts revisited. *Operations Research Letters*, 19(1):1–9, 1996.
- [125] Panos M Pardalos and Mauricio GC Resende. *Handbook of applied optimization*. 2002.
- [126] Harlan Crowder, Ellis L Johnson, and Manfred Padberg. Solving large-scale zero-one linear programming problems. *Operations Research*, 31(5):803–834, 1983.
- [127] Krasimira Genova and Vassil Guliashki. Linear integer programming methods and approaches—a survey. *Journal of Cybernetics and Information Technologies*, 11(1), 2011.
- [128] Jason Lee. Network optimization using linear programming and regression. Master’s thesis, Robert D. Clark Honors College, 2016.
- [129] T. Mastelic, W. Fdhila, I. Brandic, and S. Rinderle-Ma. Predicting resource allocation and costs for business processes in the cloud. In *2015 IEEE World Congress on Services*, pages 47–54, June 2015.

-
- [130] Arthur HM ter Hofstede, Wil MP van der Aalst, Michael Adams, and Nick Russell. *Modern Business Process Automation: YAWL and its support environment*. Springer Science & Business Media, 2009.
- [131] Mathias Weske. *Business process management: concepts, languages, architectures*. Springer Publishing Company, Incorporated, 2010.
- [132] YanHua Du, PengCheng Xiong, YuShun Fan, and Xitong Li. Dynamic checking and solution to temporal violations in concurrent workflow processes. *IEEE Transactions on Systems, Man, and Cybernetics*, 41:1166–1181, 2011.
- [133] Emanuele De Angelis, Fabio Fioravanti, Maria Chiara Meo, Alberto Pettorossi, and Maurizio Proietti. Verifying controllability of time-aware business processes. In *International Joint Conference on Rules and Reasoning*, pages 103–118, 2017.
- [134] Wenjia Huai, Xudong Liu, and Hailong Sun. Towards trustworthy composite service through business process model verification. In *Ubiquitous Intelligence & Computing and 7th International Conference on Autonomic & Trusted Computing (UIC/ATC), 2010 7th International Conference on*, pages 422–427. IEEE, 2010.
- [135] Mohamed Afilal, Hicham Chehade, and Farouk Yalaoui. The human resources assignment with multiple sites problem. *International Journal of Modeling and Optimization*, 5(2):155, 2015.
- [136] Salem M Al-Yakoob and Hanif D Sherali. Mixed-integer programming models for an employee scheduling problem with multiple shifts and work locations. *Annals of Operations Research*, 155(1):119–142, 2007.
- [137] C Mas Machuca, Jiajia Chen, and Lena Wosinska. Pon protection architectures achieving total cost reduction. In *Asia Communications and Photonics Conference and Exhibition*, pages 707–708. IEEE, 2010.
- [138] Mengshi Lu, Lun Ran, and Zuo-Jun Max Shen. Reliable facility location design under uncertain correlated disruptions. *Manufacturing & Service Operations Management*, 17(4):445–455, 2015.
- [139] Luis Jesús Ramón Stroppi, Omar Chiotti, and Pablo David Villarreal. Extended resource perspective support for bpmn and bpel. In *CIbSE*, pages 56–69, 2012.
- [140] Jianrui Wang and Akhil Kumar. A framework for document-driven workflow systems. *Business Process Management*, 3649:285–301, 2005.
- [141] Online. BPMN2 Modeler. <https://www.eclipse.org/bpmn2-modeler/>.

-
- [142] Parastoo Mohagheghi, Wasif Gilani, Alin Stefanescu, Miguel A Fernandez, Bjørn Nordmoen, and Mathias Fritzsche. Where does model-driven engineering help? experiences from three industrial cases. *Software & Systems Modeling*, 12(3):619–639, 2013.
- [143] Günther R Raidl and Jakob Puchinger. Combining (integer) linear programming techniques and metaheuristics for combinatorial optimization. In *Hybrid metaheuristics*, pages 31–62. Springer, 2008.
- [144] M. Laguna and J. Marklund. *Business Process Modeling, Simulation and Design*. CRC Press, 2013.
- [145] Mehdi Ahmed-Nacer, Slim Kallel, Faiez Zalila, Philippe Merle, and Walid Galoul. Model driven simulation of elastic occi cloud resources. Technical report, Telecom SudParis, France, 2019.
- [146] Andreas Lanz, Manfred Reichert, and Barbara Weber. A formal semantics of time patterns for process-aware information systems. Technical report, Universität Ulm, 2013.
- [147] Carlos Arévalo, MJ Escalona, I Ramos, and M Domínguez-Muñoz. A meta-model to integrate business processes time perspective in bpmn 2.0. *Information and Software Technology*, 77:17–33, 2016.
- [148] Carlo Combi, Matteo Gozzi, Jose M Juarez, Barbara Oliboni, and Giuseppe Pozzi. Conceptual modeling of temporal clinical workflows. In *14th International Symposium on Temporal Representation and Reasoning (TIME'07)*, pages 70–81. IEEE, 2007.
- [149] Carlo Combi and Roberto Posenato. Controllability in temporal conceptual workflow schemata. In *International Conference on Business Process Management*, pages 64–79. Springer, 2009.
- [150] Carlo Combi and Roberto Posenato. Towards temporal controllabilities for workflow schemata. In *2010 17th International Symposium on Temporal Representation and Reasoning*, pages 129–136. IEEE, 2010.
- [151] 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.
- [152] Emna Hachicha, Nour Assy, Walid Gaaloul, and Jan Mendling. A configurable resource allocation for multi-tenant process development in the cloud. In *International Conference on Advanced Information Systems Engineering*, pages 558–574. Springer, 2016.

-
- [153] Imen Ben Fraj, Yousra Bendaly Hlaoui, and Leila Jemni Ben Ayed. A specification and execution approach of flexible cloud service workflow based on a meta model transformation. In *ICEIS (2)*, pages 467–473, 2017.
- [154] Kenji Watahiki, Fuyuki Ishikawa, and Kunihiko Hiraishi. Formal verification of business processes with temporal and resource constraints. In *Systems, Man, and Cybernetics (SMC)*, pages 1173–1180, 2011.
- [155] Huaqing Wang and Qingtian Zeng. Modeling and analysis for workflow constrained by resources and nondetermined time: An approach based on petri nets. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 38(4):802–817, 2008.
- [156] Kazuhiro Ogata, Thapana Chaimanont, and Min Zhang. Formal modeling and analysis of time-and resource-sensitive simple business processes. *Journal of Information Security and Applications*, 31:23–40, 2016.
- [157] Saoussen Cheikhrouhou, Nesrine Chabouh, Slim Kallel, and Zakaria Maamar. Formal specification and verification of cloud resource allocation using timed petri-nets. In *International Conference on Model and Data Engineering*, pages 40–49. Springer, 2018.
- [158] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Carolyn Talcott. *All about maude-a high-performance logical framework: how to specify, program and verify systems in rewriting logic*. Springer-Verlag, 2007.
- [159] Daniel Jackson. *Software abstraction (revised edition)*, 2012.
- [160] JianQiang Li, YuShun Fan, and MengChu Zhou. Timing constraint workflow nets for workflow analysis. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 33(2):179–193, 2003.
- [161] JianQiang Li, YuShun Fan, and MengChu Zhou. Performance modeling and analysis of workflow. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 34(2):229–242, 2004.
- [162] Wil MP Van der Aalst. The application of petri nets to workflow management. *Journal of circuits, systems, and computers*, 8(01):21–66, 1998.
- [163] Boudewijn F van Dongen, Monique H Jansen-Vullers, HMW Verbeek, and Wil MP van der Aalst. Verification of the sap reference models using epc reduction, state-space analysis, and invariants. *Computers in Industry*, 58(6):578–601, 2007.

-
- [164] BF Van Dongen, Jan Mendling, and WMP Van Der Aalst. Structural patterns for soundness of business process models. In *2006 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC'06)*, pages 116–128. IEEE, 2006.
- [165] 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.
- [166] Wasim Sadiq and Maria E Orlowska. On correctness issues in conceptual modeling of workflows. In *Proceedings of the European conference on information systems*, pages 1–22, Cork, Ireland, 1997.
- [167] Wil MP Van der Aalst. Verification of workflow nets. In *International Conference on Application and Theory of Petri Nets*, pages 407–426. Springer, 1997.
- [168] Moe Thandar Wynn, HMW Verbeek, Wil MP van der Aalst, Arthur HM ter Hofstede, and David Edmond. Business process verification—finally a reality! *Business Process Management Journal*, 15(1):74–92, 2009.
- [169] Yaqiong He, Guanjun Liu, Dongming Xiang, Jiaquan Sun, Chungang Yan, and Changjun Jiang. Verifying the correctness of workflow systems based on workflow net with data constraints. *IEEE Access*, 6:11412–11423, 2018.
- [170] Frank Puhlmann and Mathias Weske. Using the π -calculus for formalizing workflow patterns. In *International Conference on Business Process Management*, pages 153–168. Springer, 2005.
- [171] Walid Gaaloul, Sami Bhiri, and Mohsen Rouached. Event-based design and runtime verification of composite service transactional behavior. *IEEE transactions on services computing*, 3(1):32–45, 2010.
- [172] Robin Milner. *Communicating and mobile systems: the pi calculus*. Cambridge university press, 1999.
- [173] Robert Kowalski and Marek Sergot. A logic-based calculus of events. In *Foundations of knowledge base management*, pages 23–55. Springer, 1989.
- [174] Kais Klai, Samir Tata, and Jörg Desel. Symbolic abstraction and deadlock-freeness verification of inter-enterprise processes. In *International Conference on Business Process Management*, pages 294–309. Springer, 2009.
- [175] Jana Koehler, Giuliano Tirenni, and Santhosh Kumaran. From business process model to consistent implementation: A case for formal verification methods. In *Proceedings. Sixth International Enterprise Distributed Object Computing*, pages 96–106. IEEE, 2002.

-
- [176] Xiang Fu, Tevfik Bultan, and Jianwen Su. Analysis of interacting bpm web services. In *Proceedings of the 13th international conference on World Wide Web*, pages 621–630. ACM, 2004.
- [177] José Luis Díaz, Joaquín Entrialgo, Manuel García, Javier García, and Daniel Fernando García. Optimal allocation of virtual machines in multi-cloud environments with reserved and on-demand pricing. *Future Generation Computer Systems*, 71:129–144, 2017.
- [178] Jin Song Dong, Yang Liu, Jun Sun, and Xian Zhang. Verification of computation orchestration via timed automata. In *International Conference on Formal Engineering Methods*, pages 226–245. Springer, 2006.
- [179] Erich Mikk, Yassine Lakhnech, Michael Siegel, and Gerard J Holzmann. Implementing statecharts in promela/spin. In *Proceedings. 2nd IEEE Workshop on Industrial Strength Formal Specification Techniques*, pages 90–101. IEEE, 1998.
- [180] Qingtian Zeng, Huaiqing Wang, Dongming Xu, Hua Duan, and Yanbo Han. Conflict detection and resolution for workflows constrained by resources and non-determined durations. *Journal of Systems and Software*, 81(9):1491–1504, 2008.
- [181] Gianluigi Greco, Antonella Guzzo, and Domenico Saccà. A logic-based formalism to model and analyze workflow executions. In Johann Eder and Tatjana Welzer, editors, *Proceedings of the 15th Conference on Advanced Information Systems Engineering*, volume 74 of *CEUR Workshop Proceedings*, Klagenfurt/Velden, Austria, 2003. CEUR-WS.org.
- [182] Wil MP Van Der Aalst. Three good reasons for using a petri-net-based workflow management system. In *Information and Process Integration in Enterprises*, pages 161–182. Springer, 1998.
- [183] Wil MP Van Der Aalst and Arthur HM Ter Hofstede. Verification of workflow task structures: A petri-net-based approach. *Information systems*, 25(1):43–69, 2000.
- [184] Jonathan Lee and Lein F Lai. A high-level petri nets-based approach to verifying task structures. *IEEE Transactions on Knowledge and Data Engineering*, 14(2):316–335, 2002.
- [185] Volker Gruhn and Ralf Laue. Using timed model checking for verifying workflows. *Computer Supported Activity Coordination*, 2005:75–88, 2005.
- [186] Elisabetta De Maria, Angelo Montanari, and Marco Zantoni. An automaton-based approach to the verification of timed workflow schemas. In *Thirteenth International Symposium on Temporal Representation and Reasoning (TIME'06)*, pages 87–94. IEEE, 2006.

-
- [187] Johann Eder, Euthimios Panagos, and Michael Rabinovich. Time constraints in workflow systems. In *Advanced information systems engineering*, pages 286–300. Springer, 1999.
- [188] Olivera Marjanovic and Maria E Orłowska. On modeling and verification of temporal constraints in production workflows. *Knowledge and Information Systems*, 1(2):157–192, 1999.
- [189] Olivera Marjanovic. Dynamic verification of temporal constraints in production workflows. In *Proceedings 11th Australasian Database Conference. ADC 2000 (Cat. No. PR00528)*, pages 74–81. IEEE, 2000.
- [190] Hai Zhuge, To-yat Cheung, and Hung-Keng Pung. A timed workflow process model. *Journal of Systems and Software*, 55(3):231–243, 2001.
- [191] Hai Zhuge. Component-based workflow systems development. *Decision Support Systems*, 35(4):517–536, 2003.
- [192] Nabil R Adam, Vijayalakshmi Atluri, and Wei-Kuang Huang. Modeling and analysis of workflows using petri nets. *Journal of Intelligent Information Systems*, 10(2):131–158, 1998.
- [193] Peter YH Wong and Jeremy Gibbons. A relative timed semantics for bpmn. *Electronic Notes in Theoretical Computer Science*, 229(2):59–75, 2009.
- [194] Claudio Bettini, X Sean Wang, and Sushil Jajodia. Temporal reasoning in workflow systems. *Distributed and Parallel Databases*, 11(3):269–306, 2002.
- [195] Xiaoxian Yang, Tao Yu, and Huahu Xu. A novel framework of using petri net to timed service business process modeling. *International Journal of Software Engineering and Knowledge Engineering*, 26(04):633–652, 2016.
- [196] Luis E Mendoza Morales, Carlos Monsalve, and Mónica Villavicencio. Formal verification of business processes as timed automata. In *2017 12th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–6. IEEE, 2017.
- [197] Sara Rachidi, Edouard Leclercq, Yoann Pigné, and Dimitri Lefebvre. Pn modeling of discrete event systems with temporal constraints. In *2017 21st International Conference on System Theory, Control and Computing (ICSTCC)*, pages 70–75. IEEE, 2017.
- [198] José Luis Pereira and João Varajão. The temporal dimension of business processes-dealing with time constraints. *Procedia computer science*, 121:1034–1038, 2017.

-
- [199] Deng Zhao, Walid Gaaloul, Wenbo Zhang, Chunsheng Zhu, and Zhangbing Zhou. Formal verification of temporal constraints for mobile service-based business process models. *IEEE Access*, 6:59843–59852, 2018.
- [200] FDR2 User Manual. Failures-divergence refinement.
- [201] Sea Ling and H. Schmidt. Time petri nets for workflow modelling and analysis. In *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics. 'cybernetics evolving to systems, humans, organizations, and their complex interactions' (cat. no.0, volume 4, pages 3039–3044 vol.4, Oct 2000*.
- [202] Vladimir A Bashkin and Irina A Lomazova. Soundness of workflow nets with an unbounded resource is decidable. In *PNSE+ ModPE*, pages 61–75, 2013.
- [203] Kees Van Hee, Alexander Serebrenik, Natalia Sidorova, and Marc Voorhoeve. Soundness of resource-constrained workflow nets. In *International Conference on Application and Theory of Petri Nets*, pages 250–267. Springer, 2005.
- [204] Natalia Sidorova and Christian Stahl. Soundness for resource-constrained workflow nets is decidable. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(3):724–729, 2012.
- [205] Vladimir A Bashkin and Irina A Lomazova. Resource equivalence in workflow nets. *Proc. of Concurrency, Specification and Programming (CS&P'2006). Humboldt Universitat zu Berlin*, 1:80–91, 2006.
- [206] Giray Havur, Cristina Cabanillas, Jan Mendling, and Axel Polleres. Resource allocation with dependencies in business process management systems. In *International Conference on Business Process Management*, pages 3–19. Springer, 2016.
- [207] Jiacun Wang, William Tepfenhart, and Daniela Rosca. Emergency response workflow resource requirements modeling and analysis. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 39(3):270–283, 2009.
- [208] Lars M Kristensen, Soren Christensen, and Kurt Jensen. The practitioner's guide to coloured petri nets. *International Journal on Software Tools for Technology Transfer (STTT)*, 2(2):98–132, 1998.
- [209] Wenbo Zhou, Lei Liu, Shuai Lü, and Peng Zhang. Toward formal modeling and verification of resource provisioning as a service in cloud. *IEEE Access*, 7:26721–26730, 2019.

-
- [210] Marcello M Bersani, Domenico Bianculli, Schahram Dustdar, Alessio Gambi, Carlo Ghezzi, and Srđan Krstić. Towards the formalization of properties of cloud-based elastic systems. In *Proceedings of the 6th International Workshop on Principles of Engineering Service-Oriented and Cloud Systems*, pages 38–47. ACM, 2014.
- [211] Kais Klai and Hanen Ochi. Model checking of composite cloud services. In *2016 IEEE International Conference on Web Services (ICWS)*, pages 356–363. IEEE, 2016.
- [212] Ali Rezaee, Amir Masoud Rahmani, Ali Movaghar, and Mohammad Teshnehlab. Formal process algebraic modeling, verification, and analysis of an abstract fuzzy inference cloud service. *The Journal of Supercomputing*, 67(2):345–383, 2014.
- [213] Hamza Sahli, Faiza Belala, and Chafia Bouanaka. A brs-based approach to model and verify cloud systems elasticity. *Procedia Computer Science*, 68:29–41, 2015.
- [214] Alessio Gambi, Antonio Filieri, and Schahram Dustdar. Iterative test suites refinement for elastic computing systems. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, pages 635–638. ACM, 2013.
- [215] Mohamed Graiet, Amel Mammam, Souha Boubaker, and Walid Gaaloul. Towards correct cloud resource allocation in business processes. *IEEE Transactions on Services Computing*, 10(1):23–36, 2016.
- [216] Hwai-jung Hsu and Feng-jian Wang. An incremental analysis for resource conflicts to workflow specifications. *Journal of Systems and Software*, 81(10):1770–1783, 2008.
- [217] Jingfu Zhong and Binheng Song. Verification of resource constraints for concurrent workflows. In *Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC'05)*, pages 8–pp. IEEE, 2005.
- [218] Qingtian Zeng, Cong Liu, and Hua Duan. Resource conflict detection and removal strategy for nondeterministic emergency response processes using petri nets. *Enterprise Information Systems*, 10(7):729–750, 2016.
- [219] Carlos Arévalo Maldonado, Isabel Ramos Román, and María José Escalona Cuaresma. Discovering business models for software process management—an approach for integrating time and resource perspectives from legacy information systems. In *Proceedings of the 17th International Conference on Enterprise Information Systems*, pages 353–359, 2015.

-
- [220] Fairouz Fakhfakh. A refinement-based approach for verifying dynamic changes on time-aware processes. *Procedia Computer Science*, 159:1489–1498, 2019.
- [221] Fairouz Fakhfakh, Hatem Hadj Kacem, and Ahmed Hadj Kacem. Towards a formal approach for verifying dynamic workflows in the cloud. In *European, Mediterranean, and Middle Eastern Conference on Information Systems*, pages 144–157. Springer, 2018.
- [222] Fairouz Fakhfakh, Hatem Hadj Kacem, and Ahmed Hadj Kacem. Dealing with structural changes on provisioning resources for deadline-constrained workflow. *The Journal of Supercomputing*, 73(7):2896–2918, 2017.
- [223] Haoyu Luo, Xiao Liu, Jin Liu, Bo Han, and Yun Yang. Adaptive temporal verification and violation handling for time-constrained business cloud workflows. In *International Conference on Service-Oriented Computing*, pages 90–99. Springer, 2018.
- [224] Haoyu Luo, Xiao Liu, Jin Liu, and Yun Yang. Propagation-aware temporal verification for parallel business cloud workflows. In *2017 IEEE International Conference on Web Services (ICWS)*, pages 106–113. IEEE, 2017.
- [225] Yeguo Wang, Rongbin Xu, Futian Wang, Haoyu Luo, Menglong Wang, and Xiao Liu. Sliding-window based propagation-aware temporal verification for monitoring parallel cloud business workflows. In *2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD))*, pages 449–454. IEEE, 2018.
- [226] QingTian Zeng, FaMing Lu, Cong Liu, Hua Duan, and ChangHong Zhou. Modeling and verification for cross-department collaborative business processes using extended petri nets. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(2):349–362, 2015.
- [227] Cristina Cabanillas, José María García, Manuel Resinas, David Ruiz, Jan Mendling, and Antonio Ruiz-Cortés. Priority-based human resource allocation in business processes. In *International Conference on Service-Oriented Computing*, pages 374–388. Springer, 2013.
- [228] Weidong Zhao, Liu Yang, Haitao Liu, and Ran Wu. The optimization of resource allocation based on process mining. In *International Conference on Intelligent Computing*, pages 341–353. Springer, 2015.
- [229] Rui Han, Moustafa M Ghanem, Li Guo, Yike Guo, and Michelle Osmond. Enabling cost-aware and adaptive elasticity of multi-tier cloud applications. *Future Generation Computer Systems*, 32:82–98, 2014.

- [230] Yacine Kessaci, Nouredine Melab, and El-Ghazali Talbi. A pareto-based genetic algorithm for optimized assignment of vm requests on a cloud brokering environment. In *2013 IEEE congress on evolutionary computation*, pages 2496–2503. IEEE, 2013.
- [231] Justin Patrick Jackson. *Constrained Task Assignment and Scheduling on Networks of Arbitrary Topology*. PhD thesis, University of Michigan, 2012.
- [232] Subodha Kumar, Kaushik Dutta, and Vijay Mookerjee. Maximizing business value by optimal assignment of jobs to resources in grid computing. *European Journal of Operational Research*, 194(3):856–872, 2009.
- [233] 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.
- [234] Walter J Gutjahr and Marion S Rauner. An aco algorithm for a dynamic regional nurse-scheduling problem in austria. *Computers & Operations Research*, 34(3):642–666, 2007.
- [235] Sophia Daskalaki, Theodore Birbas, and Efthymios Housos. An integer programming formulation for a case study in university timetabling. *European Journal of Operational Research*, 153(1):117–135, 2004.
- [236] Jiajie Xu, Chengfei Liu, and Xiaohui Zhao. Resource planning for massive number of process instances. In *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*, pages 219–236. Springer, 2009.
- [237] Jia Yu, Rajkumar Buyya, and Chen Khong Tham. Cost-based scheduling of scientific workflow applications on utility grids. In *First International Conference on e-Science and Grid Computing (e-Science'05)*, pages 8–pp. Ieee, 2005.
- [238] Ehab Nabil Alkhanak, Sai Peck Lee, and Saif Ur Rehman Khan. Cost-aware challenges for workflow scheduling approaches in cloud computing environments: Taxonomy and opportunities. *Future Generation Computer Systems*, 50:3–21, 2015.
- [239] Saeid Abrishami, Mahmoud Naghibzadeh, and Dick HJ Epema. Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds. *Future Generation Computer Systems*, 29(1):158–169, 2013.
- [240] AR Arunarani, D Manjula, and Vijayan Sugumaran. Task scheduling techniques in cloud computing: A literature survey. *Future Generation Computer Systems*, 91:407–415, 2019.

-
- [241] Xiumin Zhou, Gongxuan Zhang, Jin Sun, Junlong Zhou, Tongquan Wei, and Shiyuan Hu. Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based heft. *Future Generation Computer Systems*, 93:278–289, 2019.
- [242] Yongkui Liu, Lihui Wang, Xi Vincent Wang, Xun Xu, and Lin Zhang. Scheduling in cloud manufacturing: state-of-the-art and research challenges. *International Journal of Production Research*, 57(15-16):4854–4879, 2019.
- [243] B Muni Lavanya and C Shoba Bindu. Systematic literature review on resource allocation and resource scheduling in cloud computing. *International Journal of Advanced Information Technology (IJAIT) Vol. 6*:2231–1920, 2016.
- [244] P.J. Assudani and Satheesh Abimannan. Cost efficient resource scheduling in cloud computing: a survey. *International Journal of Engineering and Technology(UAE)*, 7:38–43, 10 2018.
- [245] Om Sangwan and Monika Dhanda. Qos based scheduling techniques in cloud computing: Systematic review. *International Journal of Computer Science and Information Security*, 07 2019.
- [246] Shweta Varshney and Sarvpal Singh. A survey on resource scheduling algorithms in cloud computing. *International Journal of Applied Engineering Research*, 13(9):6839–6845, 2018.
- [247] Long Thai, Blesson Varghese, and Adam Barker. A survey and taxonomy of resource optimisation for executing bag-of-task applications on public clouds. *Future Generation Computer Systems*, 82:1–11, 2018.
- [248] Maria Alejandra Rodriguez and Rajkumar Buyya. Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds. *IEEE Transactions on Cloud Computing*, 2(2):222–235, 2014.
- [249] Mohsen Amini Salehi and Rajkumar Buyya. Adapting market-oriented scheduling policies for cloud computing. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 351–362. Springer, 2010.
- [250] Saeid Abrishami and Mahmoud Naghibzadeh. Deadline-constrained workflow scheduling in software as a service cloud. *Scientia Iranica*, 19(3):680–689, 2012.
- [251] Ulrich Lampe, Melanie Siebenhaar, Ronny Hans, Dieter Schuller, and Ralf Steinmetz. Let the clouds compute: cost-efficient workload distribution in infrastructure clouds. In *International Conference on Grid Economics and Business Models*, pages 91–101. Springer, 2012.

- [252] Arkaitz Ruiz-Alvarez and Marty Humphrey. Toward optimal resource provisioning for cloud mapreduce and hybrid cloud applications. In *Proceedings of the 2014 IEEE/ACM International Symposium on Big Data Computing*, pages 74–82. IEEE Computer Society, 2014.
- [253] Aly Megahed, Ahmed Nazeem, Peifeng Yin, Samir Tata, Hamid Reza Motahari Nezhad, and Taiga Nakamura. Optimizing cloud solutioning design. *Future Generation Computer Systems*, 91:86–95, 2019.
- [254] Aly Megahed, Mohamed Mohamed, and Samir Tata. A stochastic optimization approach for cloud elasticity. In *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, pages 456–463. IEEE, 2017.
- [255] Sharrukh Zaman and Daniel Grosu. A combinatorial auction-based mechanism for dynamic vm provisioning and allocation in clouds. *IEEE Transactions on Cloud Computing*, 1(2):129–141, 2013.
- [256] Pratik P Pandya and Hitesh A Bheda. Dynamic resource allocation techniques in cloud computing. *International journal of advance research in computer science and management studies*, 2(1), 2014.
- [257] Xiaoxi Zhang, Zhiyi Huang, Chuan Wu, Zongpeng Li, and Francis Lau. Online auctions in iaas clouds: Welfare and profit maximization with server costs. *ACM SIGMETRICS Performance Evaluation Review*, 43(1):3–15, 2015.
- [258] Laiping Zhao, Liangfu Lu, Zhou Jin, and Ce Yu. Online virtual machine placement for increasing cloud provider’s revenue. *IEEE Transactions on Services Computing*, 10(2):273–285, 2015.
- [259] Gruia Calinescu, Amit Chakrabarti, Howard Karloff, and Yuval Rabani. An improved approximation algorithm for resource allocation. *ACM Transactions on Algorithms (TALG)*, 7(4):48, 2011.
- [260] Takfarinas Saber, James Thorburn, Liam Murphy, and Anthony Ventresque. Vm reassignment in hybrid clouds for large decentralised companies: A multi-objective challenge. *Future Generation Computer Systems*, 79:751–764, 2018.
- [261] Johan Tordsson, Rubén S Montero, Rafael Moreno-Vozmediano, and Ignacio M Llorente. Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers. *Future generation computer systems*, 28(2):358–367, 2012.
- [262] Chahrazed Labba, Nour Assy, Narjès Bellamine Ben Saoud, and Walid Gaaloul. Adaptive deployment of service-based processes into cloud federations. In *International Conference on Web Information Systems Engineering*, pages 275–289. Springer, 2017.

- [263] Fahimeh Ramezani, Jie Lu, and Farookh Hussain. Task scheduling optimization in cloud computing applying multi-objective particle swarm optimization. In *International Conference on Service-oriented computing*, pages 237–251. Springer, 2013.
- [264] Haluk Topcuoglu, Salim Hariri, and Min-you Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE transactions on parallel and distributed systems*, 13(3):260–274, 2002.
- [265] Aly Megahed, Kugamoorthy Gajananan, Mari Abe, Shun Jiang, Mark Smith, and Taiga Nakamura. Pricing it services deals: A more agile top-down approach. In *International Conference on Service-Oriented Computing*, pages 461–473. Springer, 2015.
- [266] Kugamoorthy Gajananan, Aly Megahed, Mari Abe, Taiga Nakamura, and Mark Smith. A top-down pricing algorithm for it service contracts using lower level service data. In *2016 IEEE international conference on services computing (SCC)*, pages 720–727. IEEE, 2016.
- [267] Sumanta Basu, Soumyakanti Chakraborty, and Megha Sharma. Pricing cloud services—the impact of broadband quality. *Omega*, 50:96–114, 2015.
- [268] Shadi Ibrahim, Bingsheng He, and Hai Jin. Towards pay-as-you-consume cloud computing. In *Proceedings of the IEEE International Conference on Services Computing*, pages 370–377. IEEE Computer Society, 2011.
- [269] Sahar Arshad, Saeed Ullah, Shoab Ahmed Khan, M. Daud Awan, and M. Sikan-dar Hayat Khayal. A survey of cloud computing variable pricing models. In *Proceedings of the 10th International Conference on Evaluation of Novel Approaches to Software Engineering*, pages 27–32. SciTePress, 2015.
- [270] Parnia Samimi and Ahmed Patel. Review of pricing models for grid & cloud computing. In *Proceedings of the IEEE Symposium on Computers & Informatics*, pages 634–639. IEEE, 2011.
- [271] Diego Cardoso Alves, Bruno Guazzelli Batista, Dionisio Machado Leite Filho, Maycon Leone Maciel Peixoto, Stephan Reiff-Marganiec, and Bruno Tardiole Kuehne. CM cloud simulator: A cost model simulator module for cloudsims. In *Proceedings of the IEEE World Congress on Services*, pages 99–102. IEEE Computer Society, 2016.
- [272] Evert Duipmans. *Business process management in the cloud: business process as a service (BPaaS)*. PhD thesis, University of Twente, 2012.
- [273] MingXue Wang, Kosala Yapa Bandara, and Claus Pahl. Process as a service distributed multi-tenant policy-based process runtime governance. In *Proceedings*

- of the *IEEE International Conference on Services Computing*, pages 578–585. IEEE, 2010.
- [274] Mehdi Ahmed-Nacer, Kunal Suri, Mohamed Sellami, and Walid Gaaloul. Simulation of configurable resource allocation for cloud-based business processes. In *Proceedings of the IEEE International Conference on Services Computing*, pages 305–313. IEEE, 2017.
- [275] Mathias Weske. Business process management architectures. In *Business Process Management*, pages 333–371. Springer, 2012.
- [276] Saoussen Cheikhrouhou, Slim Kallel, Nawal Guermouche, and Mohamed Jmaiel. The temporal perspective in business process modeling: a survey and research challenges. *Service Oriented Computing and Applications*, 9(1):75–85, 2015.
- [277] Maja Pesic, M. H. Schonenberg, Natalia Sidorova, and Wil M. P. van der Aalst. Constraint-based workflow models: Change made easy. In Robert Meersman and Zahir Tari, editors, *Proceedings of the International Conferences on the Move to Meaningful Internet Systems*, volume 4803 of *Lecture Notes in Computer Science*, pages 77–94. Springer, 2007.
- [278] Wenjia Huai, Xudong Liu, and Hailong Sun. Towards trustworthy composite service through business process model verification. In *Proceedings of the 2010 7th International Conference on Ubiquitous Intelligence and Computing and 7th International Conference on Autonomic & Trusted Computing*, pages 422–427. IEEE Computer Society, 2010.
- [279] Waheed Ahmad, Robert de Groote, Philip K. F. Hölzenspies, Mariëlle Stoelinga, and Jaco van de Pol. Resource-constrained optimal scheduling of synchronous dataflow graphs via timed automata. In *14th International Conference on Application of Concurrency to System Design, ACSD 2014, Tunis La Marsa, Tunisia, June 23-27, 2014*, pages 72–81, 2014.
- [280] Qiang Li and Yike Guo. Optimization of resource scheduling in cloud computing. In *International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, 2010.
- [281] Menglan Hu, Jun Luo, and Bharadwaj Veeravalli. Optimal provisioning for scheduling divisible loads with reserved cloud resources. In *18th IEEE International Conference on Networks*, pages 204–209. IEEE, 2012.
- [282] Daniel Solow. Linear and nonlinear programming. *Wiley Encyclopedia of Computer Science and Engineering*, 2007.

-
- [283] Marcello La Rosa, Wil MP Van Der Aalst, Marlon Dumas, and Fredrik P Milani. Business process variability modeling: A survey. *ACM Computing Surveys (CSUR)*, 50(1):1–45, 2017.
- [284] Marcello La Rosa, Petia Wohed, Jan Mendling, Arthur HM Ter Hofstede, Hajo A Reijers, and Wil MP van der Aalst. Managing process model complexity via abstract syntax modifications. *IEEE Transactions on Industrial Informatics*, 7(4):614–629, 2011.
- [285] Occiware. <https://www.occware.org/>, (Mai 20, 2019).
- [286] Gifty Gupta and Neeraj Mangla. Workflow scheduling in cloud computing. *Journal of Computational and Theoretical Nanoscience*, 16(9):3965–3968, 2019.