Bournemouth University

FACULTY OF SCIENCE AND TECHNOLOGY

Supporting Compliance Verification for Collaborative Business Processes

Author:

John Paul KASSE

This dissertation is submitted for the degree of Doctor of Philosophy

August 2019

Abstract

Collaborative business processes are the current trend of business processes supported by the advances in technology like the Internet and collaborative networks. Enterprises no longer do business in isolation. The customer demands are always changing and becoming sophisticated with dynamic requirements and the shortening period in which they must be met. Collaborative business processes must conform with not only customer demands but also with laws, standards, best practice and regulations. These impose constraints on the business process that must be satisfied otherwise they attract criminal charges or financial fines. Corporate scandals for companies like Enron, Worldcom, Societe General etc. were a result of non-compliance. This attracted regulations like the Sarbanese Oxley Act, Basel III, Anti money laundering act among others with articles guiding operational practice.

However, non compliance is still observed especially among SMEs that do not possess the skilled man power or the funding to acquire automated compliance solutions. In this thesis, we sought to support non-expert end users through a compliance management approach that can guide the specification and verification of compliance for collaborative business process with a range of policy and regulatory requirements. Collaborative business processes differ from traditional business processes. They are characterised by specific attributes that present unique verification requirements that cannot be automatically addressed by existing verification approaches. To achieve the intended goal, design science research method was employed to develop a mechanism to elicit requirements from different sources, translate them into formal constraints based on formal semantics, and a set of algorithms were composed to support compliance verification. The algorithms provide meaningful and easy to understand feedback to the end user about the compliance or violation of the collaborative business process.

Due to the fact that policies and regulations change often, we adopted simulation analysis as a technique to assess and analyse the impact of such changes to the business process before actual implementation.

The thesis artifacts are evaluated based on known information systems model evaluation methods following the design science recommended steps and the Method Evaluation model (MEM). We also validate and evaluate the compliance algorithms using a different industrial use case (the car insurance trading business process) from the case used in their design (the pick and pack business process). Further more, the performance of the algorithms is evaluated based on their computation complexity.

Dedication

"If you want to lift yourself up, lift up someone else."

Booker T. Washington

To everyone who has given me a hand, been patient with me and has prayed for me. More especially, My wife Ann and children Jean Paula, Jordin Jordan Paul, Jayryn Privy, Justice Penniela, Judge Providence, Johanna Price, Jeremiah Pricey.

Declaration

I declare that this thesis is my work. The pronouns 'we' and 'our' are employed for flow and style purposes.

Acknowledgement

"Success is no accident. It is hard work, perseverance, learning, studying, sacrifice and most of all, love of what you are doing or learning to do."

Pele

I am profoundly grateful to God for the opportunity, life and protection, strength and courage, knowledge and wisdom. Him alone knows the end from the beginning. Glory be to his name.

Secondly, sincere appreciation to my first supervisor Dr. Lai Xu for the professional guidance, academic advice and encouragement. Thank you for accepting to work with me right from the time I made the first contact with you, and thank you for your patience. Great appreciation to Dr. de Vrieze for the continuous guidance and technical help. Thanks to Prof. Keith Phalp for the support. The opportunity accorded to work on the EU FIRST Project was such a rewarding experience from which I gained knowledge, exposure and new networks in Germany and China.

To my wife Ann and the children, thank you for the love, support, encouragement and patience for the time we missed together. The future hold better for us all. To my sisters, Flavia, Joan, Mary and uncle Henry. Thanks to Collin for the support. Thank you very much for helping me start life in the UK. To my Mum Mary, thank you for the love, prayers and encouragement. To My friends with whom we shared the office (P319), it was a great opportunity to meet you and the experiences we shared. All the best in your future endeavours. My fellowship Friends Tim and Sang, we bless the Lord for the journey we worked. Thanks for the prayers and courage. Lastly but not least, with gratitude i thank the Principal Prof. Waswa Balunywa and MUBS management for contributing to the matched fund and sponsorship.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

”If you define the problem correctly, you almost have the solution.”

Steve Jobs
Apple Inc

1.1 Background

Business process management (BPM) brings together knowledge from computer science and management science to support the design, management and implementation of business processes through application of techniques, methods and tools [3]. Through the various suggested methodologies, management of business processes is enforced through design and analysis, configuration, enactment and execution, implementation and monitoring. In other words, a business process undergoes a life cycle. The BPM life cycle is composed of three overlapping phases, i.e. (Re) design and analysis, implement/configure and, Run and Adjust.

The (Re) design phase refers to the design and analysis of business processes. Models of business processes are created and analysed at design time. The implement/configure phase transforms the business process model into a running system (e.g. a process aware information system) while in the Run and Adjust phase the process models are implemented, run and adjusted as may be required. The data logged during the execution is used to support further analysis and verification of the running processes. Figure 1.1 illustrates the phases of the life cycle. At the core of BPM is the business process upon

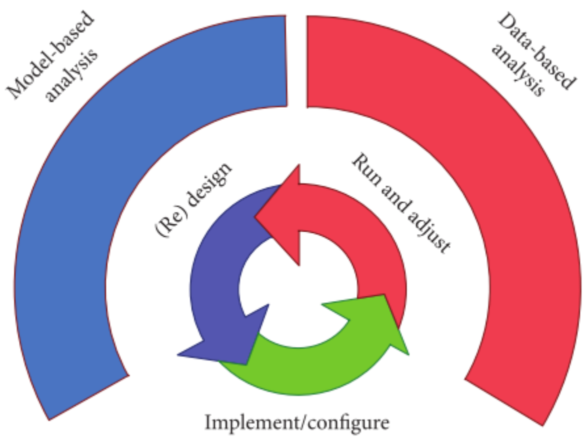


Figure 1.1: BPM life cycle [3]

which organisations activities are modelled, organised and managed [44].

To understand the business process, various definitions have been used;

- a structured set of logically related activities performed to achieve a business outcome [146].
- adds value for the customer by processing inputs into outputs [66].
- a structured set of activities designed to produce a specific output [42].

Given the various definitions, for common understanding in this report the business process is defined as;

A structured set of interrelated activities routinely performed within and between organization(s) to achieve a predetermined outcome.

The adopted definition considers structured business processes within and among organizations. This implies that;

- Business processes are also unstructured and informal. However, formal and structured processes are preferred to support application of formal methods and tools for design and analysis.

- Business processes exist internally (traditional business processes) or among different organizations (collaborative business processes). The interest of this research concentrates on the latter type of business processes.

The rest of the chapter is structured as follows; Section 1.2 introduces the collaborative business processes and describes how they differ from traditional business processes while section 1.3 presents the methods for composing process models. In section 1.4, the subject of business process verification is introduced forming a basis for understanding business process compliance which is introduced in section 1.5 and process compliance verification described in section 1.6. Section 1.7 presents the technical foundation that motivated the research as well as the major research question upon which the research focus, major research objective and the specific objectives are based and stated in section 1.8. Section 1.9 lists the publications and contributions from this research while Section 1.10 presents the structure in which the thesis is presented, and section 1.11 summarises the Chapter.

1.2 Collaborative Business Processes (CBP)

Besides the fact that collaborative business processes are conducted among several organisations to achieve a common business goal [27], they are characterised by specific attributes not common to traditional business processes. CBP are described by complex dynamic behaviour which is considered to autonomous yet collaborative, distributed yet interrelated, stable yet dynamic [121]. Collaborative business processes are defined as processes that cross organisational borders where multiple organisations or several partners operate on shared business process [9].

To describe and represent business processes, models are designed following process modelling principles [159]. Models present a diagrammatical expression of business processes to facilitate their understanding, analysis and improvement. Various tools are used for this purpose based on formalisms like petri nets, Event Chains, UML and business process management notation (BPMN). Chapter 2 provides extended analysis of these formalisms and modelling tools. However, in this thesis all models are presented based on BPMN for simplicity and convenience of illustrations. For example, Figures 1.2 and 1.3 show two BPMN models illustrating the disparity between traditional and collaborative business processes respectively. In these figures, the same process is modelled from two perspectives i.e. as a traditional single organisation business process (Figure 1.2) or as an interactive collaborative business process executed among several participants.

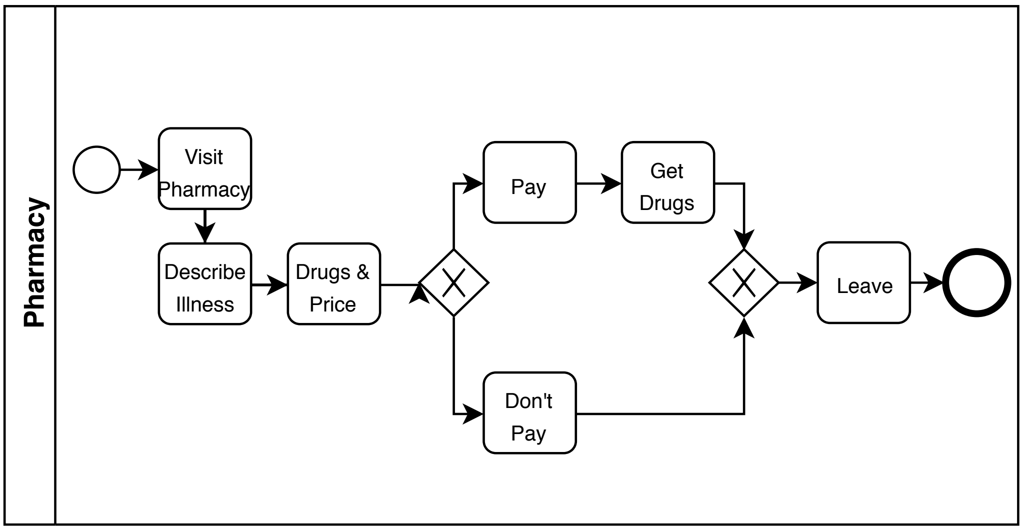


Figure 1.2: Single organisation business process

In Figure 1.2 model, a patient can buy drugs if the illness requires no prescription. The patient enters the pharmacy, describes the illness to the pharmacist who will identify necessary drugs, asks the patient to pay, issue the drugs and the patient leaves the pharmacy. If the patient cannot pay for some reasons, no drugs are issued. Whereas in Figure 1.3, the patient must have a prescription in order to acquire drugs or get treatment. In this scenario, the business process model represents collaboration between the patient, hospital, pharmacy and Insurance Company. The patient visits the general practitioner (GP) for diagnosis. After, the GP prescribes drugs or other treatment, the patient visits a pharmacy to acquire drugs. The pharmacist will receive the prescription, identify drugs and ask the patient to pay. The patient can pay by cash or use insurance cover if she/he is a policy holder. If the patient does not pay, the process ends. Otherwise the drugs

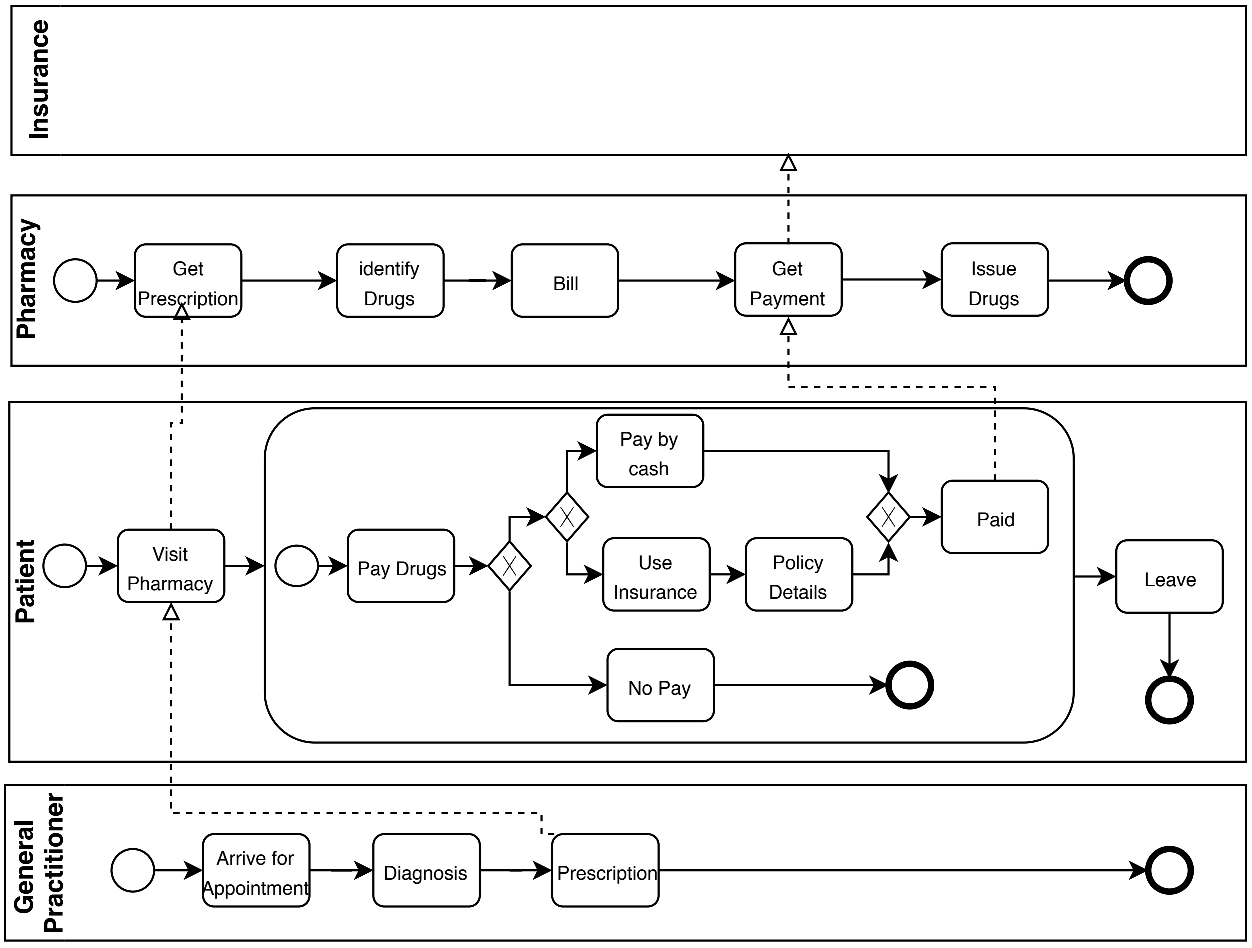


Figure 1.3: a collaborative business process

are issued after payment and the patient exits the pharmacy. If the patient was a policy holder, the pharmacy claims the payment from the insurance company.

1.3 Composing Models

There are different ways through which process models are designed. For instance, models are designed from scratch, discovered from event log data, selected from existing models, by merging different model parts or by combining individual models into one bigger model. The outcome from modelling process are models which are; descriptive (e.g. as-is model or as-to-be model situations), normative (i.e. representing the logical sequence of activities) or executable (interpretable by tools and systems) [3].

Due to the structural differences between the traditional and collaborative business processes, the designing of collaborative business processes presents a unique challenge; the process must satisfy a set of business and operational requirements from all stakeholders, conform with regulatory requirements in form of policies, laws and standards regulations as a cross organisation or cross border process. Moreover, distinctive characteristics impose unique requirements necessary for the design and specification of collaborative business processes; the need to keep organisation specific data private, creation and formal specification of partner interfaces for interaction, mapping them to executable processes, need to support data flow between partners, various cross organisational units and roles that must be supported, need for semantic annotation synchronisation [171] Ziemann and Matheis 2007, Lippe and Ziemann 2011 increase the complexity of collaborative business processes necessitating the need for continuous checking and verification [86].

1.4 Business Process Verification

Despite being described as normative, models may exhibit undesirable behaviour in form of errors that can prevent successful execution or lead to undesired behaviour at runtime. Errors like incompleteness, inconsistencies and ambiguities if unchecked are passed on from design and specification to actual business processes. It is therefore necessary to verify models through diagnosis, identification and checking for such errors. Process model verification encompasses the identification of errors inherent in models [163, 164] at either design time, runtime or both. Verification is a way to prove that the designed process possesses required properties and at the same time does not have bugs. Formal verification involves the application of mathematical models to prove correctness of a design given a set of specifications. To achieve verification, three main techniques are used [18]:

– *Theory proving*: a technique for verifying systems by formally constructing and checking derivations using formulas, axioms and inference rules for deriving new formulas form existing ones. Theory proving applies logic like high order Logic (HOL) to reason about artifacts.

– *Simulation*: a technique used to study and analyse properties concerning model behaviour which facilitates understanding the actual process behaviour.

– *Model checking*: Figure 1.4 shows model checking verification technique. In this technique models are verified by exploring all of their states systematically. The techniques check the conformance of the model behaviour specified as a system against a set of properties specified as constraints. Model checking employs the use of temporal logic to support reasoning and discovery of errors that may escape simulation.

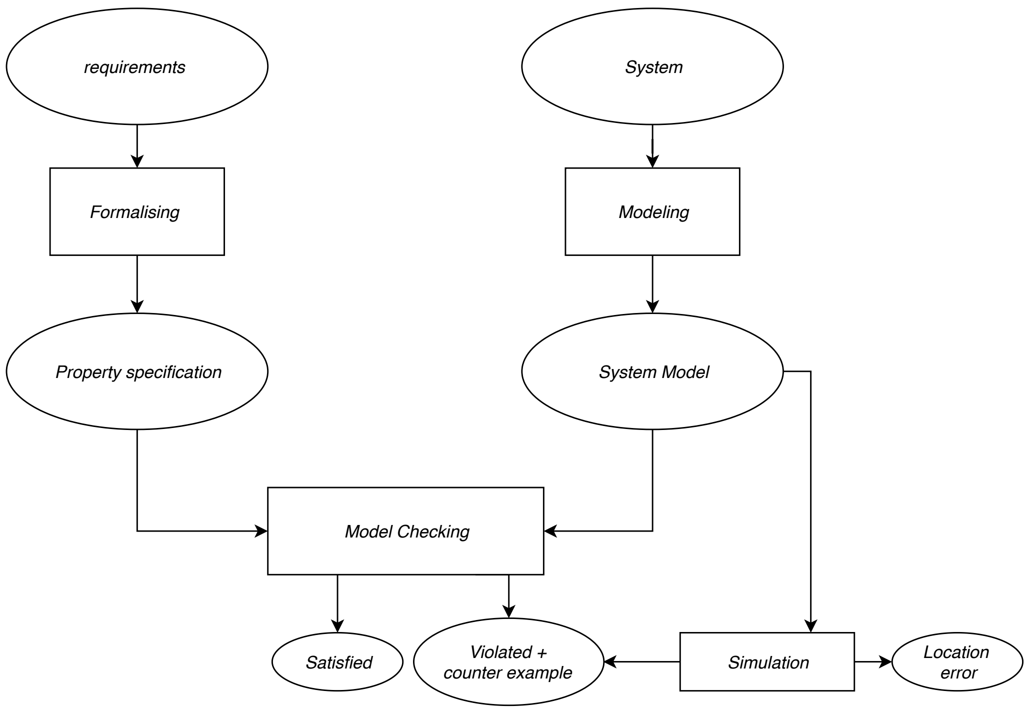


Figure 1.4: a view of model checking adopted from [18]

Figure 1.4 illustrates the steps undertaken to achieve model checking. The preliminary steps involve establishing requirements to form properties to be checked, formalising them and translating them into formal properties through specification. On the other hand, the system to be checked is specified through modelling into a system model which then is model checked against the properties. The outcome would then show whether the properties are satisfied, if not, a counter example is given. The advantage of model checking over other methods is the ability to point to the source of the error. In this thesis, a combination of simulation and model checking techniques is used are used to support verification of collaborative business processes and checking them for compliance.

1.5 Business Process Compliance

Compliance is a big topic in the business world today and costing the industry huge sums of money and time in fines and litigation or tuning the business processes to comply with the requirements of standards, laws and regulations [130, 86]. Due to the corporate world financial scandals involving giant companies like Enron, Tyco, Global Crossing and Adelphia, Enron, HIH, Société Générale, WorldCom inter alia [83], compliance has come to the fore as a measure to guide and monitor corporate business behaviour world to avoid a repeat. Strict regulations and laws were instituted for not only finance industry but to all sectors world over, e.g. the Sarbanes-Oxley (SOX) Act of 2002 [139], the Basel III Accord [19], HIPPA (HIPAA 2018) and the consumer protect Act 2015 (UK), National Institute of Standards and Technology (NIST), ITIL, ISO/IEC 20000 (international standard for IT service management), System and Organisation Controls (SOC). The regulatory agencies monitor organizations to ensure that they compliant to the regulations specified. Such regulations are external to organisations and are observed on top of internal policies within the organisation which are established to guide business operations. From an organizational perspective therefore, compliance must be achieved for both internal and external policies and regulations.

Compliance refers to adherence to rules, norms, laws and other regulatory requirements like standards and best practice. In terms of business processes, compliance has been defined in several ways as follows;

"A desired outcome, with regard to law and regulations, internal policies and procedures, and commitment to stakeholders that can be consistently achieved through managed investment of time and resources. Compliance management includes the legal and tactical activities in day to day business processes" [104].

"The relationships between the formal specifications of a business process and the formal specifications of a set of normative constraints, where a process is compliant if the specifications of the processes do not violate the constraints formalising the norms" [59].

"an act or process to ensure that business operations, processes, and practices are in accordance with prescriptive (often legal) documents" (Governatori and Scientific 2014) "as ensuring that business processes, operations, and practice are in accordance with a prescribed and/or agreed set of norms" [130].

"as the process of ascertaining the adherence of business processes and applications to relevant compliance requirements, which may emerge from laws, legislation, regulations, standards and code of practices (such as ISO 9001), internal policies, and business partner contracts [46].

Whereas the first definition provides a generic definition of compliance, the second, third and fourth definitions consider compliance in terms of processes being in accordance with legal documents. The fifth definition is specific to adherence to compliance requirements. For the benefit of this thesis, business process compliance is defined as; A set of activities undertaken to ensure adherence of business processes to specific internal and external regulatory requirements throughout its life cycle.

The new definition considers compliance as continuous process throughout the life cycle of a business process i.e. design time, runtime and post-run-time. This is so because policies and regulations change over time, so should the new or existing business processes relatively comply with the changes. To achieve regulatory compliance of existing and new collaborative business process necessitates checking their behaviour at different phases of their life cycle. However, checking and verifying compliance is not automatic more especially where collaborative business processes are involved. As discussed in section 1.3, collaborative business processes have more requirements to comply with; organisation specific requirements, contractual obligations and other external regulations. Compliance management involves a set of activities that compose its life cycle; elicit, formalise, implement, check and improve compliance requirements of a specific regulation. Figure 1.5 illustrates the phases over the compliance continuum.

Besides compliance life cycle management being part of the organisation strategies, it also plays a central role to ensure that other strategies comply with relevant policies and regulations. For instance, the software development strategy has to comply with the software development standards and practices. Figure 1.6 presents the relationship between compliance life cycle management and other organisational strategies.

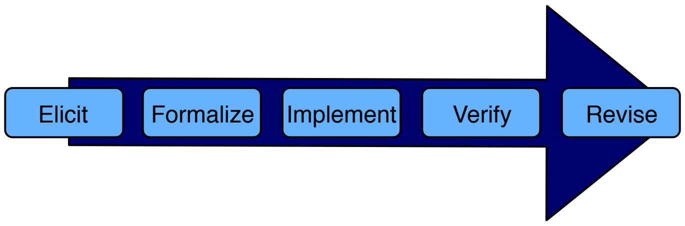


Figure 1.5: Compliance life cycle

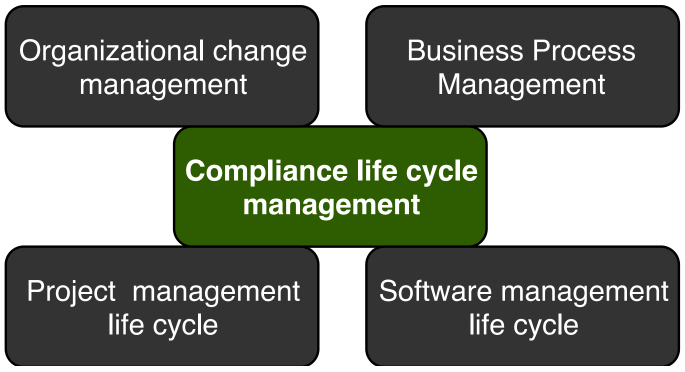

Organizational change management
Business Process Management
Compliance life cycle management
Project management life cycle
Software management life cycle

Figure 1.6: Relationship of BPM and compliance

1.6 Compliance Verification for Business Processes

Business process verification involves various strategies; design time, runtime and post runtime. These strategies also apply to compliance checking and verification. *Design time verification* is a preventive approach that aims at checking a process's compliance with requirements during its design. Design time verification is a detective approach often preferred for early time identification of errors leading to non-compliance or violation. Whereas, *runtime verification* strategy checks process compliance during its execution. *Post execution verification* is an auditing activity involving manual procedures to check process compliance after its execution. The hybrid approach combines all or more than one of the verification strategies. Figure 1.7 illustrates compliance verification strategies According to Hashmi et al., several dimensions of compliance checking compose the business process compliance space [67]. As Figure 1.7 illustrates, the compliance space has also been used to cast the research trend and state of art in compliance verification for business processes. In this thesis, attention is paid to a hybrid strategy leading to a compliance verification approach for collaborative business processes to facilitate checking of existing processes against new regulatory requirements or against

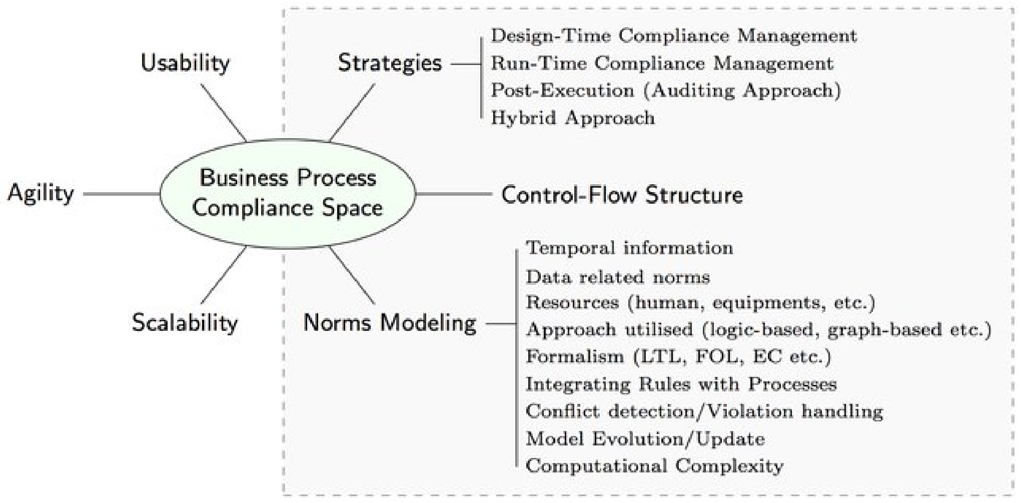

Usability
Strategies
Design-Time Compliance Management
Run-Time Compliance Management
Post-Execution (Auditing Approach)
Hybrid Approach
Agility
Business Process Compliance Space
Control-Flow Structure
Scalability
Norms Modeling
Temporal information
Data related norms
Resources (human, equipments, etc.)
Approach utilised (logic-based, graph-based etc.)
Formalism (LTL, FOL, EC etc.)
Integrating Rules with Processes
Conflict detection/Violation handling
Model Evolution/Update
Computational Complexity

Figure 1.7: Business Process Compliance Space [67]

changes in regulatory requirements. In a detailed fashion, we show how to support end users to verify processes in a compliance continuum in Chapter 8.

1.7 Motivation and Research Question

The current trend of business organisations shows a paradigm shift from closed business environments to border-less interconnected ones [27, 25, 26, 9]. This further termed as de-parameterisation where business processes are jointly offered to meet the dynamic demands from service consumers especially where traditional single organisation business processes cannot offer satisfactory services. This trend is enabled by technological advancements like internet, cloud computing, service-oriented computing and e-commerce. Collaborative business processes have consequently emerged. As earlier stated, collaborative business processes have to comply with more regulatory requirements from various sources, and involve multi-partners. Achieving compliance at this magnitude is complex especially for non-expert end users. In this work we postulate that it is necessary to support such end users in order to realise compliant processes since they are the subject matter experts. This leads to the question that this research seeks to address.

How to support end users verify collaborative business processes for policy and regulatory compliancy?

The research is motivated by the need to support the verification of compliance of collaborative business processes with policy and regulatory requirements where multi parties are involved as opposed to the traditional single party business processes. As the related work in Chapter 2 indicates, most of the existing work like, [135, 56, 132, 39, 60, 142, 52] address compliance verification situated in non-collaborative environments. Moreover, compliance solutions in the industry are proprietary with a high level of rigidity, limited interoperability and lack flexibility when applied to different environments or different compliance requirements [46]. Others require high expertise from the end users with proven skills that ordinary end-users do not possess. Small scale businesses cannot afford proprietary solutions let alone the required skills to operate and use these solutions. Despite efforts undertaken in compliance and its enforcement, non-compliance and violations of regulations is still on the rise in the economic, medical, software and social industries. This could be attributed to a sea of regulatory requirements that organisations cannot keep track with.

Moreover, changes and amendments in policies and regulations come frequently from the regulators. Such changes directly impact on the local policies which drive business operations. This implies that organisations have to check and review local policies and business policies fast enough to match the regulatory demands before assigned deadlines to avoid fines or litigation. For example, the European Union amended the general data protection regulation (GDPR) with a deadline for May 2018. The amendments cost organisations time, resources and money to achieve compliance. Financial Times reported a survey by Ernst and Young that implementing the requirements from the revised GDPR would cost top companies up to $ 7.8 billion [89]. In another survey by Veritas technologies, it was suggested that companies were likely to spend an average of € 1.3 million on systems, hiring new staff and training to comply with the GDPR [41].

There is need for a mechanism to support verifying of amended regulatory requirements with existing organisational business processes. Moreover, it is useful to have *a priori* assessment of the likely impact of the changes of policies and regulations to the process's structure, resources and data so that modellers can get knowledge to inform their decisions by analysing several scenarios before actual implementations. Based on the outcome, recommendations, changes or reviews can be made, or use outcome to modify the processes to achieve compliance. A mechanism to support this kind of compliance verification is lacking. An integrated flexible solution is required to meet a set of requirements for a compliance verification framework. Such requirements include;

1. To support the checking and verification of compliance of existing business processes with amended policies and regulations. This is achieved by a mechanism that supports identification of compliance violations at different levels of checking. Segments of the business process affected by the amendments and verifying them against modified policies. Afterwards the entire process is verified to ensure that the entire business process remains compliant.

2. To support end users to design and verify collaborative business processes through less complex compliance checking algorithms. The thesis presents different compliance checking algorithms that have been designed respectively. Combinations between different algorithms could lead to different compliance checks.

3. To Support business process optimisation based on changes in the policies and regulations through simulation based analysis of different use case scenarios and provide recommendations to support informed decision making. The simulation should as well support the generation of traces from designed processes for supporting process compliance checks.

1.8 Research Focus and Objectives

Following from Figure 1.7, the focus of this research is a hybrid strategy leading to support for continuous compliance checking of collaborative business process with regulatory requirements at any of the various phases of the life cycles of both business process management and compliance management. With a hybrid compliance checking approach, it implies that a process can be checked from scratch e.g. during design time or specific checking against a specific change in policy or regulations, or checking for compliance to a particular constraint. We show the elicitation, formalisation and checking of the compliance constraints against the business process at different levels of the life cycle. Using activity events and process instances, a business process is broken down into execution traces from which checking is conducted using the composed algorithms put forward in Chapter 8.

From the compliance life cycle, the study addresses phases in which compliance requirements are elicited from source documents to their formalisation. The logical relationship between process design and compliance checking is enforced by mapping the compliance requirements on to the process model and verifying for compliancy of the model to the specified requirements through application of formal model checking techniques. The research presents a mechanism based on description logic through which users can elicit and check compliance of processes without need for rigorous mathematics.

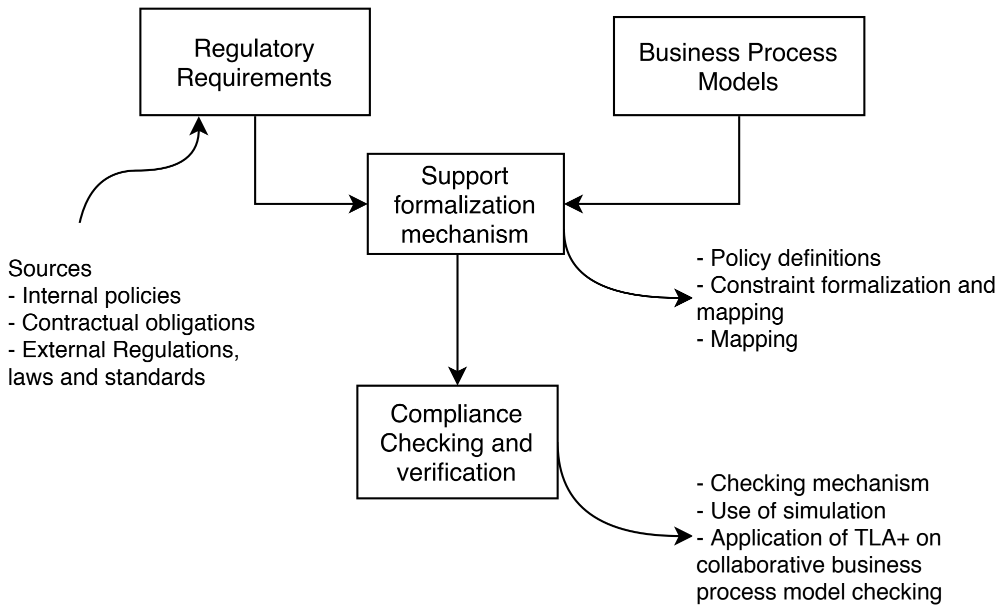

Regulatory Requirements
Business Process Models
Support formalization mechanism
Sources
- Internal policies
- Contractual obligations
- External Regulations, laws and standards
- Policy definitions
- Constraint formalization and mapping
- Mapping
Compliance Checking and verification
- Checking mechanism
- Use of simulation
- Application of TLA+ on collaborative business process model checking

Figure 1.8: Research Focus

Figure 1.8 summarises the research focus while Figure 1.9 presents the research scope. Compliance requirements are elicited from policy and regulatory documents, formalised into constraints. While the business process are checked for compliance with verification constraints using the verification algorithms. Feedback is provided once violations are detected . Based on the research question and focus,the major objective of the study is derived as follows;

To develop an approach that supports end users verify collaborative business processes for compliance with policy and regulatory requirements.

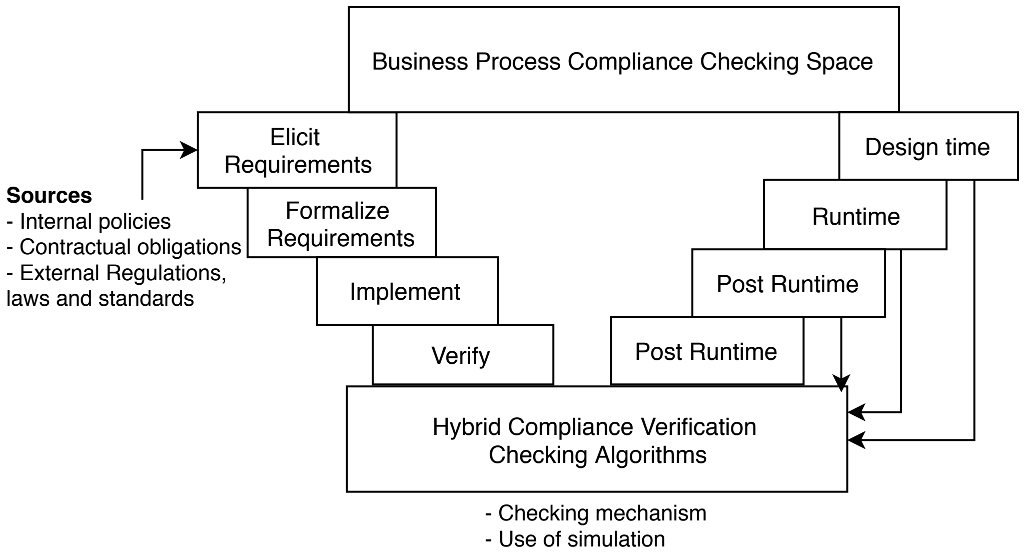


Figure 1.9: Research Scope

To achieve the desired outcome, more specific objectives are derived from major objective as stated below;

1. To support the elicitation and translation of compliance requirements from source documents into compliancy constraints.

2. To demonstrate the application of simulation and analysis as a technique to support:

    (a) Assessment of the impact of policy and regulatory variations over existing, new or re-designed business processes.

    (b) Generation of traces from the new or re-designed business processes for compliance verification.

3. Design a compliance verification framework for supporting compliance verification of collaborative business processes with policy and regulatory requirements from control flow, data, resource and temporal perspectives through application of algorithms that;

    (a) Detect compliance violations.

    (b) Report on the status of compliance.

(c) Apply process driven authorisation and access control (PDAC), a novel mechanism for implementing privacy and authentication access control requirements.

4. Evaluate the designed algorithms using industry based use cases.

5. Propose architectures for practical implementation of PDAC and the overall collaborative business process compliance verification approach.

1.9 Research Dissemination

In addition to this thesis, the research has been disseminated through presentations at various workshops and conferences as well publications in conference proceedings and journals. The listed publications have been accepted, presented and published in conference proceedings while the journal paper is pending acceptance. The publications are listed below with brief description of their contents.

Paper 1

Kasse, J.P., Xu, L. and de Vrieze, P., 2017, September. A Comparative Assessment of Collaborative Business Process Verification Approaches. In Working Conference on Virtual Enterprises (pp. 355-367). Springer, Cham.

Paper 2

Kasse, J.P., Xu, L. and Bai, Y., 2018, September. The Need for Compliance Verification in Collaborative Business Processes. In Working Conference on Virtual Enterprises (pp. 217-229). Springer, Cham.

Paper 3

Kasse, J.P., Xu, L. and Bai, Y., 2019, February. Process Driven Access Control and Authorisation Approach. In Fourth International Congress on Information and Communication Technology (pp…). Springer, Singapore.

Paper 4

Shuangyu Wei, Yuewei Bai, Xiaogang Wang, Liu Kai, Lai Xu, Paul de Vrieze and John Paul Kasse: A New Method for Manufacturing Process Autonomous Planning in Intelligent Manufacturing System. 1st International Workshop on Key Enabling Technologies for Digital Factories. CAiSE 2019. June 2019, Rome, Italy.

Paper 5

Kasse, J.P., Xu, L., de Vrieze, L and Bai, Y., 2019. Verifying Compliance with Data Constraints for Business Processes: Collaborative Networks and Digital Transformation. 23 - 25 September 2019 – Turin, Italy.

Journal Paper - Pending

Compliance Verification for Collaborative Business Processes. International Journal of Simulation and Process Modelling.

H2020 FIRST Project Deliverables

1. D1.2: overview of service-oriented business process verification
2. D1.3: Overview of existing interoperability of virtual factories
3. D2.1: Manufacturing Asset/Service description languages
4. D4.1: On-the-fly Service Oriented Process verification and implementation

1.10 Structure of the Thesis

The thesis is composed of ten (10) interlinked chapters as Figure 1.10 illustrates.

In *chapter 1*, an introduction of the thesis is presented including a background to the discipline in which research is situated, an explanation of the motivation, highlight of issues concerning business processes, compliancy and the need for verifying business processes for compliancy with regulations, standards and policies. Further, the research question which this research sought to address is presented upon which scope and objectives that guided the study were set. The chapter ends with research contributions and publications. The rest of the thesis is organised as follows.

Chapter 2 presents the related work to this thesis starting with a technical and theoretical background to the key variables and concepts to the study like policies, regulations, requirements and constraints categorisation inter alia. Further, analysis of existing solutions is presented while pointing out limitations that informed the propositions to our contribution. *Chapter 3* presents the research methodology followed to accomplish the goals of the research. A description of details about the method and tools employed is described.

Chapter 4 presents detailed discussion about policies, their formulations, elicitation from general regulations to local policies. Further, the chapter presents the use cases that are used for demonstration throughout the thesis. Simulation as a verification technique is presented demonstrating the impact of policy variations over business processes. A Bizagi simulation tool is used for demonstrations.
Chapter 5 presents constraints and requirements definition based on Description Logic language. Based on the use case, we show how to extract compliancy requirements and present them in a manner easy to understand by non-expert end users.

Chapter 6 provides the translation and formalisation of extracted constraints into linear temporal logic to facilitate reasoning over them for verification purposes. In this way, formalised constraints are mapped with the process model and checked for compliancy.
Chapter 7 presents the verification approach to support compliancy verification between constraints and business process model. Several sub algorithms are presented for verification and checking of the model at different levels or based on the different categories of constraints.
Chapter 8 presents the compounded algorithms for both constraint checking and violation detection.
Chapter 9 is an evaluation of the verification algorithms based on the use cases and a discussion of the outcomes and observations.
Chapter 10 is the conclusion of the thesis highlighting general observations, future work and recommendations. A summary of the chapter outlines is shown in Figure 1.10.

1.11 Chapter Summary

The chapter provided a preamble to the rest of the thesis by introducing business process management and compliance management as the disciplines under which the report is situated. A background from the two disciplines is presented discussing existing challenges which motivated the study. Based on motivation the research question, objectives, goals and scope of the study are specified. Additionally, the chapter presents the contributions of the research to the field of literature in terms of conference papers published and pending. Contributions were also made to the EU FIRST project deliverable.

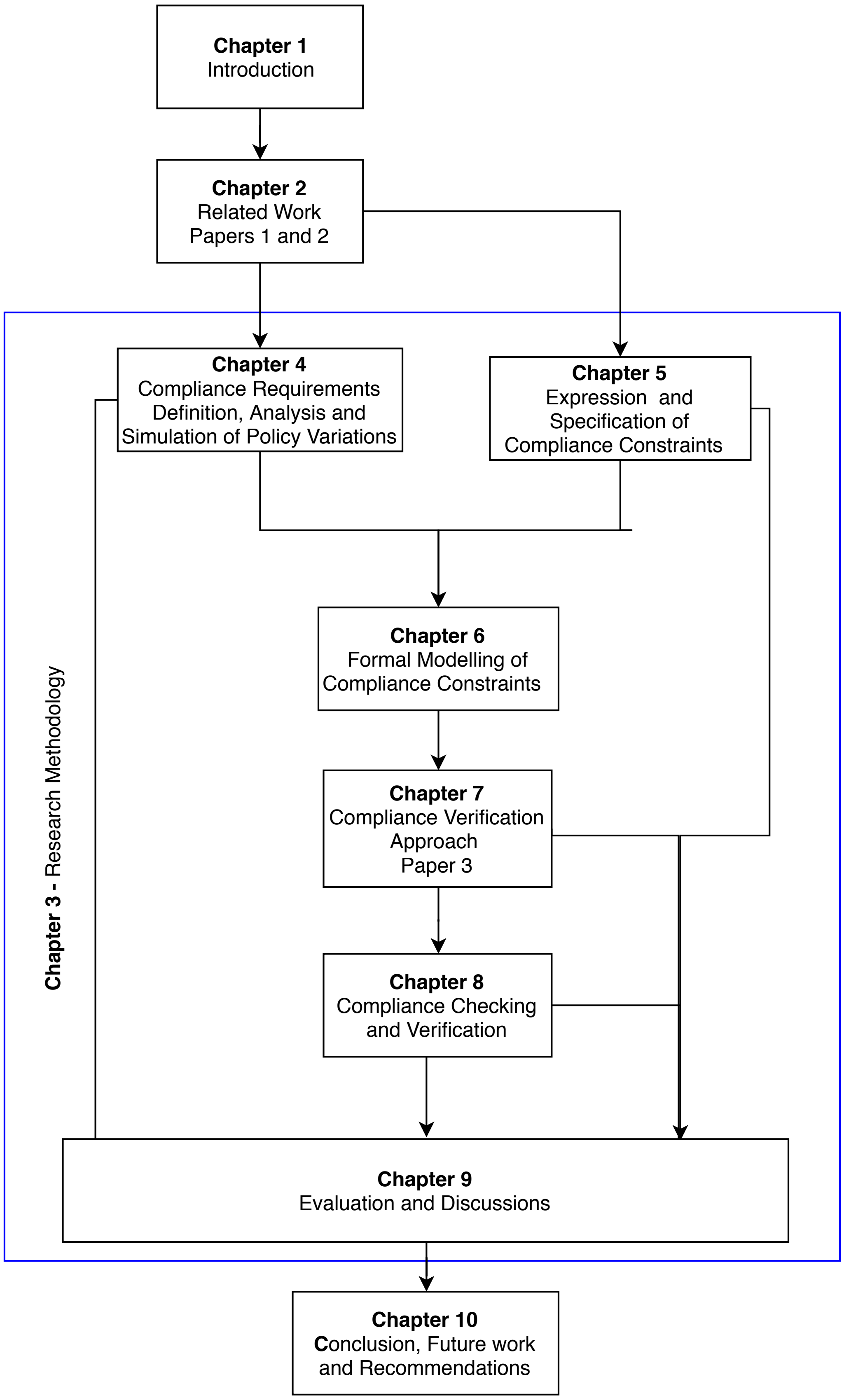


Figure 1.10: chapter Outline

Chapter 2

Related Work

"All I want is compliance with my wishes, after reasonable discussion."

Winston Churchill

2.1 Introduction

The chapter aims to explore and introduce various relevant concepts, work, potential challenges and requirements related to verification of collaborative business processes with policies and regulations. The motive is to provide understanding of the state of art offered in the various methods, techniques, frameworks and their supporting tools. The chapter is presented as follows: Section 2.2 introduces policies and regulations while section 2.3 presents policies and regulations as sources of verification requirements and constraints. Section 2.4 and 2.5 present policy implementation strategies, policy definition languages respectively while section 2.6 cites examples of policies and regulations. In section 2.7 the common compliance requirements are presented from the state of art while in section 2.8 the business process behaviour is presented. Section 2.9 categorises the different forms of business process constraints in terms of control flow, data, resource and temporal patterns. The concept of process verification is introduced in section 2.10 and, section 2.11 presents the state of the art in compliance verification. Verification requirements for collaborative business processes are presented in section 2.12 while section 2.13 summarises the verification requirements from existing research and exciting approaches for Compliance Verification. Section 2.14 presents the chapter summary.

2.2 Policies and Regulations

Policies embody action plans that guide decisions leading to logical outcomes. Cambridge online dictionary broadly defines a policy in different ways;

"A set of ideas or a plan of what to do in particular situations that has been agreed to officially by a group of people, a business organisation, a government, or a political party"

"A set of ideas or a plan for action followed by a business, a government, a political party, or a group of people"

From the definitions, a commonality that stands out is that the policy is a plan for a course of action to be done in specific cases. In this thesis we therefore define policies as norms and behaviours agreed upon by parties (business, government etc.) as a formal procedure to guide business operations. Whereas policies are internal to an organisation or its partners designed to guide its behaviour, regulations are normally external instituted to guide entire sector or industry. A regulation can be understood as a rule made and maintained by an entity which is an authority. Examples are national and international laws regulating cross border operations, e.g. national and international tax policies, national and international standards like accounting standards, best practice e.g. medical practice.

They are normally expressed as orders, directives, acts, laws, statutes, ordinances or guidelines. Hence forth, we refer to policies as internal or local rules while regulations as external rules and laws regulating operations of a business. Policies and regulations form controls that specifically restrict and constrain what should be done or what should not be done.

2.3 Policies and Regulations in Relation to Business Processes

In relation to BPM, policy and regulatory requirements form guidelines and constraints that restrict a set of permissible or forbidden behaviour over the structure and elements of an organisation's business processes. For some cases, regulations impose a legal requirement which organisations are required to comply with by law. These form mandatory requirements. Different regulations exist and present a myriad of requirements to regulate behaviour of organisations in the industry. However, for compliancy purposes, an organisation will identify requirements relevant to its business processes. Therefore, from a range of regulations and policies, an organisation will identify and select a set of requirements mandatory for compliancy. A compliance requirement is a defined extract from the general regulatory document concerning a specific regulatory guideline. A collection of the requirements from different regulations and policies form a document of relevant compliancy requirements to guide behaviour of a given business process is composed. Extraction of requirements from source documents is not enough, they have to be transformed and translated into a form that is enforceable to facilitate verification over business process models

A compliance requirement is a defined specific extract from the general regulatory document regarding a regulatory guideline. As a result, a document of relevant compliance requirements to guide behaviour of a given business process is composed from different regulations and policies. Extraction of requirements from source documents is not enough, they have to be transformed and translated into a form that is enforceable to facilitate verification over business process models.

2.4 Policy Implementations Strategies

In policy based systems, different strategies are implemented to evaluate and enforce policies at various points. Examples of such systems are security systems or app for implementing access control and authorisation. Implementation strategies adopted include;

- *Policy Decision Point (PDP)* is a point at which policies are evaluated and decisions taken concerning access and authorisation. The PDP evaluates an access request against a policy and decides whether access is permitted or denied.
- *The Policy Enforcement Point* enforces the policy decisions taken at PDP. PEP receives the access request, forwards it to the PDP, receives the decision and enforces it by permitting or denying access. Based on the concept, extra information may be required for evaluation to arrive at a decision.
- *The policy information point* provides extra external attribute information e.g. from the user registry to support request evaluation to arrive at access or deny decision.
-*The Policy Administration Point (PAP)* is a central point at which all policies are managed.

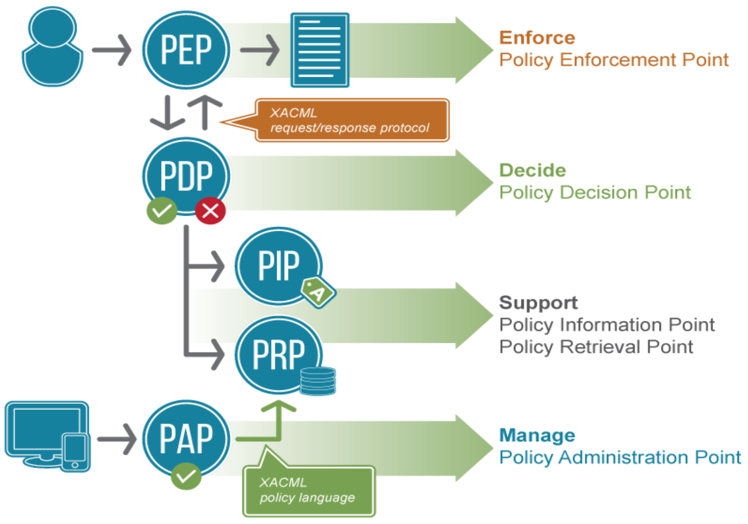


Figure 2.1: Policy Implementation Architecture based on XACML [140]

Figures 2.1 and 2.2 illustrate the policy implementation strategies and implementation architecture respectively.

2.5 Policy Definition and Specification Languages

Various languages are used to specify and define policies. Ordinarily policies are defined as documents of policy statements in human understandable language. To avoid the ambiguity associated with natural languages and support formalisation, policy defi-

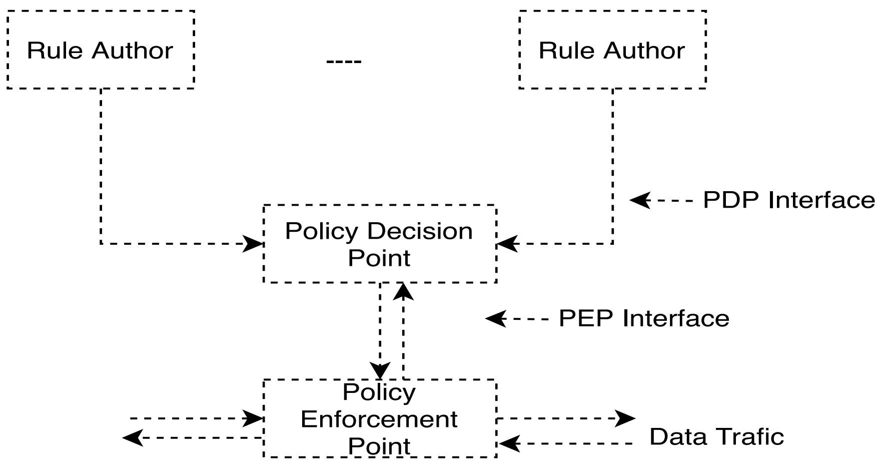


Figure 2.2: Policy Implementation strategies illustration

nition languages like *XML, XAML and SAMXL* are used to define and specify policies in computer understandable form.

- XML: Extensible Markup Language is used to define rules for encoding documents in both human and machine-readable formats with simplicity, generality, and usability across platforms. It supports message exchange to facilitate communication and interoperability between systems and facilitates exchange of data and information via the standardised XML schema.

- XAML: based on XML as a declarative language applied in initialisation of values and objects.

- SAML: Security Assertion Markup Language is an XML-based open standard for data exchange between parties or systems regarding authentication and authorisation e.g. identity provider and a service provider. Specifically, SAML supports specification of security leaning policies on which access control decisions are based. The specification defines the roles of the principal, identity provider and the service provider that make, authenticate and permit requests. These are referred to as SAML assertions expressing a packet containing security information.

2.6 Examples of Policies and Regulations

Depending on the domain, a business process is required to comply with several regulations. Different regulations present different compliance requirements, for example;

1. The General Data Protection Regulation (GDPR). The regulation from EU was revised in 2018 with the revised version emphasizing data privacy and security. The regulation specifies a set of requirements regarding personal data;
   - Must be protected from intentional or unintentional misuse.
   - Data owner must grant access to its use and must know who and where the data is being processed.
   - The regulation emphasises security by design principle where data privacy requirements is built within the system. For example, privacy aware business processes and information systems.
2. The Sarbanes-Oxley (SOX) Act of 2002 [139] (SOX 2002) and the Basel III Accord (Basel 2018) are accounting regulations that target fraud prevention or detection in business processes and workflow systems. The key requirements from the two regulations are;
   - Separation of duty i.e. two activities are executed by different resources
   - Binding of duty i.e. tasks that must be executed by the same resource.
3. The consumer protection act specifies requirements that empower customer rights and protection.
   - Right quality of goods and services
   - Right to return unsatisfactory goods or services etc.
4. International Standards Organisation (ISO) presents different forms of certifications with different requirements like 9001, 27001, 14001 among others

The cited regulations show a mix of requirements that a business process is required to comply with. Compliance is not automatic but rather a non-trivial task for end users to accomplish. Moreover, compliance is not by choice but a legal requirement. A step towards achieving compliance is to understand the compliance requirements from the regulations relevant to a particular organisation. The next section presents some of the key compliance requirements from some common regulations.

2.7 Common Compliance Requirements: State of the Art

- *Separation of duty (SoD)* : SoD is a policy and regulatory requirement that dates to decades ago. However, it is more pronounced by the SOX regulation [139]. SoD concept refers to separation of concerns for both tasks and resources assigned for their execution, where two disjoint tasks must be executed by the different resources or roles. At least more than one role is required to complete a task in a process [23, 143].

- *Binding of duty*: In contrast to SoD, BoD is requirement stipulates that execution of two or more tasks should be executed by the same resource or role [23, 143].

The requirements of SoD and BoD are rooted in the SOX and Basel III regulations and implemented as internal policies in business.

Illustration of SoD and BoD

An example of purchase order processing is used to illustrate the two requirements. A requisition for printer cartridges is raised and approved. The cartridges are delivered and goods received note is signed. The example shows three internal tasks; request, approve, received. To enforce SoD and BoD requirements, the three tasks cannot be executed by a single resource i.e. same user to writes the requisition and approve it, this is the case of SoD. To exemplify the case with roles, to enforce SoD would require the office clerk to raise the requisition which is then approved by another role e.g. the line supervisor. The duties of requisition and approval are separated and executed between the clerk and the supervisor. Similarly to enforce BoD the clerk will receive and sign for the cartridges. In this way, requisition and sign for goods received tasks are executed as BoD by the office clerk.

- *Security requirements* – Modern enterprises run enterprise information systems in which workflow systems are a component or run independent workflow systems. In SOX and Basel III regulations, SoD and BoD are specified as security requirements to prevent or detect fraud and other errors that may compromise the security of systems [110]. In workflow systems, security is a major requirement to be implemented and complied with to information security and assurance. Cherdantseva and Hilton define information security and assurance as below [36]

"Information Security ... is concerned with the development and implementation of security countermeasures of all available types (technical, organisational, human-oriented and legal) in order to keep information in all its locations (within and outside the organisation's perimeter) and, consequently, information systems, where information is created, processed, stored, transmitted and destructed, free from threats". "Information Assurance ... aims to protect business by reducing risks associated with information and information systems by means of a comprehensive and systematic management of security countermeasures, which is driven by risk analysis and cost-effectiveness"

A security policy describes the representation of rules that enable the implementer to determine if requests for access should be granted, given the values of the attributes of the subject, object, and possibly environment conditions. The Subject describes a user (human or application) identified by a set of attributes which requests access to perform operations on objects. The Object refers to a resource for which access is requested and upon which the operation will be conducted by the subject once access is granted. E.g. files, records, tasks, processes, programs, etc. While Operation: describes the actions like read, write, edit, delete, copy, execute, and modify conducted on the object by the subject. Lastly the environment includes the prevailing conditions and context in which access requests are made and granted like the current time, day of the week, location of a user, or the current threat level [79].

Cherdantseva and Hilton describe a set of requirements that systems should comply with to be secure;

- Auditability; An ability of a system to conduct persistent, non-by-passable monitoring of all actions performed by humans or machines within the system
- Authenticity/Trustworthiness: Ability of a system to verify identity and establish trust in a third party and in information it provides
- Availability; ensuring that all system's components are available and operational when they are required by authorised users
- Confidentiality; ensuring that only authorised users access information
- Integrity; ensuring completeness, accuracy and absence of unauthorised modifications in all its components
- Non-repudiation; An ability of a system to prove (with legal validity) occurrence/non-occurrence of an event or participation/non-participation of a party in an event
- Privacy; system should obey privacy legislation and it should enable individuals to control, where feasible, their personal information (user-involvement)

Since business processes must comply with the policy and regulatory requirements, their behaviour is therefore driven to behave as the requirements specify. In the section that follows, a further discussion of how the regulatory requirements relate with business process behaviour is presented. In addition, categorisation of these requirements is discussed in relation to process behaviour.

2.8 Business Process Behaviour

Business process models provide a graphical means to describe normal or expected behaviour of the business process. The core business process perspectives depict the behaviour of a business process in terms of control flow, resource flow, data flow and temporal perspectives [10]. To facilitate compliance to the requirements, a logical relation is established to relate the policy and regulatory requirements with the behavioural perspectives of the business process. Establishment of such relationship benefits process design following policies and regulations as design guidelines i.e. each process perspective is specified according to a set of rules and policies which constrain the process to execute in a specific behaviours. Furthermore, following the categorisation, it enables us to discuss the related work in the field. The process perspectives present a unique way

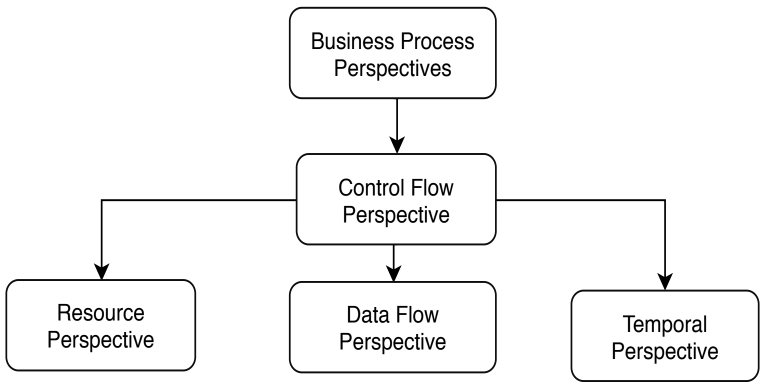


Figure 2.3: Business Process Perspectives [10]

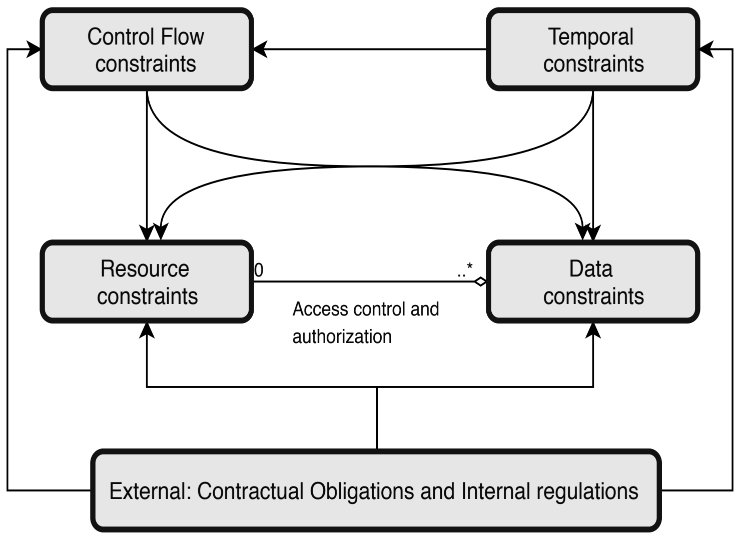

Control Flow constraints
Temporal constraints
Resource constraints
0
..*
Data constraints
Access control and authorization
External: Contractual Obligations and Internal regulations

Figure 2.4: Relationship between process perspectives

to look at the business process behaviour from a structural setup. They also present a basis upon which a logical interrelationship between the process and the constraints that determine its behaviour. Using process perspectives a relationship is established which facilitates derivation and categorisation of compliance requirements from the general policies and regulations. Figure 2.3 shows the key business process perspectives whereas figure 2.4 depicts the interrelationship between them.

Control flow perspective is considered as a basis on which other process behaviour is based. This implies that following the ordering of activities, resources are assigned; access to data is authorised as well as the timing of events and activities of the process. Upon figure 2.3 and 2.4, a category based diagram in figure 2.5 is composed to illustrate constraint categories and how they relate to the business process perspectives. Control

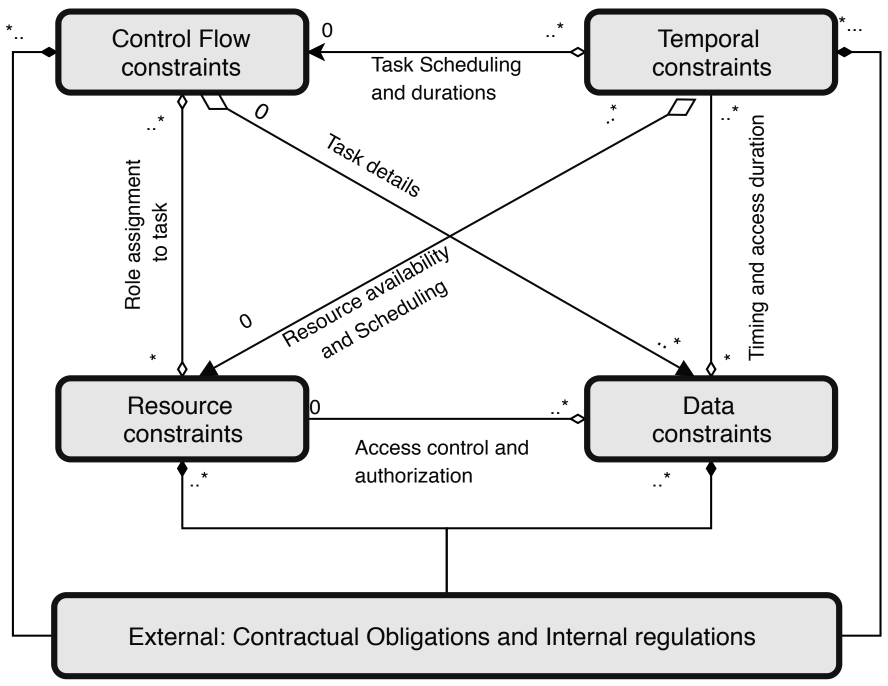

Control Flow constraints
Temporal constraints
Task Scheduling and durations
Task details
Role assignment to task
Resource availability and Scheduling
Timing and access duration
Resource constraints
Data constraints
Access control and authorization
External: Contractual Obligations and Internal regulations

Figure 2.5: Relationship between process perspectives and constraint categories

flow and resource constraints are related by assignment relationship in which resources are assigned to tasks. The relations between resource constraints and data constraints are enforced by the access control and authorisation elements. The temporal constraints are related to all constraints through scheduling activities, resources and period of access to data. Direct and transitive constraint relationships are inferred from the perspective based relationship in Figure 2.5. Direct relationships describe constraints explicitly influencing the behaviour of a process without reference to another perspective. E.g. control flow constraints directly relate to the activity, the assignment of a resource to a task. The transitive relationships implicitly define constraints whose influence on the behaviour of the process perspective is derived from another perspective. Temporal constraints are an example in this case whose effect over a task is dependent on constraints governing other perspectives. E.g. the occurrence of activity a will delay for 30 minutes, role r is scheduled to execute activity a at 10:15 hours, and access to data is authorised during day working hours between 9:00 - 17:00 hours.

2.9 Generic Constraint Patterns

Constraints are derived from policy and regulatory requirements. Control flow perspective describes the ordering of activities in the process. Thus, the constraints under this category are based on rules and policies concerning ordering of activities. Related to control flow are functional constraints for guiding activity decomposition into executable tasks and operational constraints which regulate application services. These two forms of constraint categories are concerned with constraining the execution behaviour of activities. For that matter, they are discussed under control flow. The data category includes rules and policies that govern the flow of data in and out of the process in terms of its access restrictions, control and authorisation. The resource flow constraints describe the rules and policies that restrict the assignment and allocation of resources to tasks [10, 2, 159] .

Several authors [15, 46] have used process perspectives to organise and categorise compliance requirements into patterns. Various authors have employed the use of patterns to represent common occurring permissible requirements that are used to express the behaviour of an entity to facilitate checking and verification [45]. In the same way, verification for compliance of a business process to policy and regulations is based on compliance constraint patterns.

2.9.1 Control flow Constraint Patterns

Control flow constraints restrict business processes to behave in a specific in relation to activity ordering. To this effect, this section presents control flow patterns and related work is presented. Dwyer et al (1998) propose a set of property specification patterns for finite states verification tools. The patterns are organised into hierarchies based on their semantics as Figure 2.6 illustrates.

According to Dwyer et al, users can search for a pattern matching the requirement being specified, map it into a formalism suitable for a given tool and instantiate it by plugging

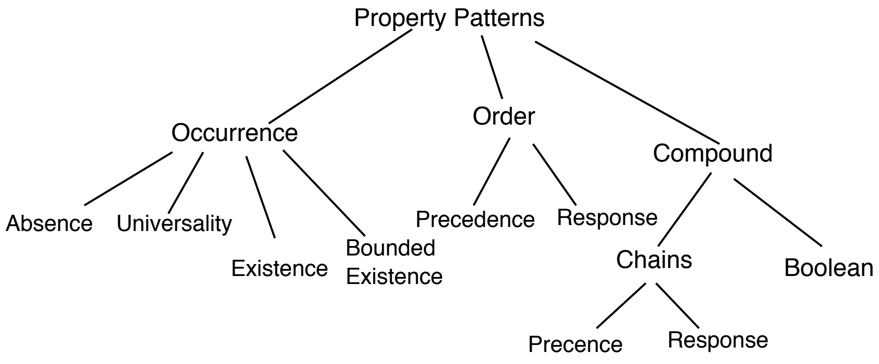


Figure 2.6: Control flow Pattern Hierarchy

the pattern into state formulas. The pattern hierarchies include Occurrence, ordering and compound as top nodes. The patterns can then facilitate model checking by verifying occurrence and ordering of an activity or chain of activities. Authors however do not address validation and verification to ensure that the right pattern is identified and its correct mapping. This forms the first requirement to be addressed in this thesis;

- ***Req1. Based on control flow constraints, verify for compliancy of collaborative business processes***

Relatedly, van der Aalst et al propose and implement control flow patterns similar to the ones above. The authors use the proposed patterns as requirements to assess the suitability and expressiveness of commercial workflows languages[10]. The patterns proposed have been very popular and induced strings of research. For example, Pesic et al. present a constraints modelling mechanism to model business processes conforming with control flow requirements [119, 117]. The authors propose a flexible approach to constraints modelling to achieve several options of model behaviour from which the designer selects the best option. Awad implements a formal approach in form of a BPMNQ tool for design time compliance checking. The tool is based on visual patterns to model compliance requirements in terms of control flow, data flow and conditional flows [15, 16, 17]. In another study in which a compliance management framework is proposed, control flow patterns are extended and formalised into a compliance request language (CRL) [46]. CRL is based on property specification patterns by Dwyer et al, van der Aalst et al and Hall et al., and expounded to include other hierarchies of resource and time patterns.

2.9.2 Data Constraint Patterns

Data in workflow systems relates to both control data required by the workflow system and process data required for task execution [10]. The former is of relevance to this discussion to derive data constraints from the policies and rules that guide operations. Data constraint patterns express the common forms of data requirements applicable in workflow systems. These are as well adopted to facilitate elicitation of compliance requirements from general policies and regulations. A number of data patterns are presented by Russel et al from a review of commercial workflow systems. These are grouped to include; data visibility, data interaction, data transfer and data-based routing [127]. In table 16 are exemplified data constraints under the categories of data visibility and data interaction which are relevant to the study. In addition, we propose other data constraints including validity, availability and accessibility. Besides, the GDPR prescribes articles for data protection in which data privacy and protection must be observed [123]. The law defines the data owner, data processor, data user and third party and stipulates their roles, responsibilities and obligations, and requires organisations to observe privacy and security by design to achieve compliancy. This way, privacy and security quality as data patterns useful to capture related compliance requirements. Authorisation and access control has emerged as a mechanism to enforce data access in workflow management systems due to rising security threats and breaches. The rise in cloud computing and its application in business process management e.g. virtualized processes, BPM as a Service (BPMaaS) and cloud storage has resulted into a paramount need for access control and authorisation. To meet the security requirements, workflow systems employee access control mechanisms that compel BPM systems and their components to comply with security requirements specified in policies. Such mechanisms are proposed in different studies [148, 155, 170, 115, 79, 110, 13].

2.9.3 Resource Constraint Patterns

Resource flow constraints restrict how resources are assigned to tasks. Well known resource patterns are separation of duty, biding of duty sources and delegation which restrict the relation between the task and its actor. SoD and BoD constraints are also implemented as controls against fraud and collusion to commit crime. Personnel resources are known to intentionally evade laws and rules for malicious intentions. Resource constraints regulate the scope of what resources can do on the tasks. When coupled with time, a resource's behaviour is restricted according temporal requirements. E.g., resource availability based on work calendar. Striking a balance between optimal resource allocations to tasks is a challenge. For example, determining the resource requirements for different tasks in consideration of resource availability in relation to when tasks are ready for execution. This, often considered an NP Hard problem [40]. Various mechanisms are used in implementation of resource constraints;

- Role based mechanism [22, 148, 172]. These mechanisms automate SoD and BoD[24, 23] administers access control by granting access to a system based on user roles in the organisation. RBAC renders a simple systematic and repeatable approach to security administration, audit and correction in case of any breach. It is premised on three key principals [50]; role assignment, role authorisation and transaction authorisation.

- Task based Access Control mechanism Task based access control models present an active authorisation management mechanism to model and specify security policies and to dynamically enforce and administer their implementation at runtime. TBAC approaches recognise the context in which the request for permissions arise and participate in security management by relating to the progress and emerging context within the tasks. Permissions are activated or deactivated, granted or revoked automatically and coordinated with task progression solving the overhead of administration [148]. The task authorisation framework presents an authorisation step formed of trustee set and Protection state. The trustee set bears the executor to authorise tasks with permissions from the executor permissions. The permissions are enabled for a period at the enabled permissions component.

- Attribute based Access Control Models. In ABAC mechanism access control and authorisation enforced by following an evaluation of attributes of the subject, object, requested operations against policy, rules, or relationships that describe the allowable operations for a given set of attributes [79], after which access is granted or denied.

2.9.4 Temporal Constraints

Temporal constraints specify the rules that regulate time requirements in a business process as established in the policy guidelines, laws and regulations. Such policies may in general specify an entire process total time or particularly specify time requirements for each activity, resource availability or data access [35]. Temporal patterns include; intervals, duration, delays and deadlines. E.g. execution intervals between tasks; $task_a$ executes for a duration of t_n units before $task_b$, period execution of a task, i.e. task start and finish times. Individual task duration aggregate into total process cycle time.
The assignment of timing relations follows absolute or relative time. In absolute allocation time values are assigned in real time limits while relative time allocates time as a single value. The challenge to achieving temporal constraints satisfaction is due to temporal uncertainty where activity duration or its instance become known only during run time [129]. Consider a scenario in which purchased goods are shipped. The duration for shipment can only be known when a shipment agent is selected or when the address of the customer becomes known at run time. Verifying for such kind of data available at run time requires runtime process monitoring. Table 5.1 represents key temporal constraints in literature [142, 122].
Against this background, it is worth to note that policy and regulations requirements and constraints guide process behaviour. It is therefore imperative to ensure conformance of the process to the expected behaviour expressed in terms of constraints that represent policy and regulatory requirements through Business process verification.

2.10 Business Process Verification

In general terms verification refers to proof of correctness of a system. In reference to BPM, process model verification refers to the means of proving existence of certain design requirements in a process model. It provides a way to prove that the intended behaviour at design time of the model is exhibited in the business process at execution

Table 2.1: Temporal constraints patterns

Constraints	Description
Instance duration	Period for which the rule must hold
Delay	Period within which an activity can be delayed
Validity	Period within which an activity can be executed
Duration	The period for which an activity is scheduled to execute form start to completion
Repetition	Period between which an activity can be repeated
Overlap	Period within which an activity can start and complete with reference to another activity's start and completion period.
Deadline	Describes the expected start or finish time of an activity

time. Verification is intended to check the connectivity of designed process models, their correctness (absence of deadlocks and live locks in processes), compensation and scalability (number of services a process model can support) and compatibility (between process variants) [109]. Best practice recommends that model errors should be identified early enough before implementation [164] . However, verification occurs at design time, execution time and post execution.

2.10.1 Design Time vs. Runtime Process Verification

Runtime Business Process, also known as on-the-fly verification involves dynamic analysis and monitoring of running processes against a set of precisely specified properties [138]. Compared to design-time verification, on-the-fly verification is scalable enough to permit analysis of evolving and executing processes. In this way, the state explosion problem is minimised. Monitoring algorithms and tools are used to characterise and specify properties which are then checked against the running systems [68, 90, 106, 47]. On the drawback, runtime verification is challenged in terms monitoring overheads. According [154], committing changes to running systems is destined to cause more errors. Moreover it's difficult to keep track of the errors, or resource wasting to correct them while the process is running. Runtime verification is applicable for verification of designs with characteristics of finite traces and inappropriate to verification of mission critical systems. It is however considered as a light weight formal way to verify designs.

Design-time Business Process Verification, also known as static analysis is conducted while the business process is being designed. It employs techniques like model checking, theorem proving and static analysis to analyse the behaviour of a system before its execution. The proof is based on the design against a set of specifications expressed and formalised using temporal logic. Model checking is known to suffer from state explosion problem limiting the scalability of the designs being verified [169], theorem proving calls for manual efforts to discover the invariants while static analysis of code is not expressive as a technique for a range of properties that can be checked [47]. Since model checking is known for verification of mission critical system designs, this research partly focuses on design-time verification of process models by means of model checking. Moreover, it is much cheaper in terms of time and effort to correct errors during model design. Furthermore, we emphasise design-time for collaborative business process models in a vF setting where little attention is realised or is lacking. The nature of verification for inter-organisation business processes differs from that of single organisation processes due to associated complexity in tracing where errors are in the models and sub models [9]. More so errors would affect the entire VE as opposed to the single organisation. To offer a scalable working solution, focus of the approach in this research is a hybrid solution that scales between design time, runtime and post execution to support compliance verification in the process life cycle. Design time compliance is achieved through compliance by design, a concept which facilitates the modelling of compliance requirements and their propagation into business processes and supporting enterprise systems [131, 130, 114].

Post Execution verification involves a manual auditing activity through which log files created during execution of a business process are archived and checked if the process conformed to required behaviour. This is done through a process known as process mining. This method is a corrective after the effect method, errors are discovered only benefit next stage of the life cycle.

Process Mining

Process mining is an effort geared towards business process discovery through automated model construction based on knowledge extracted from event logs to support process analysis and improvement. The Prom framework [43] is at the fore of hosting process mining techniques and tools classified into discovery, conformance and extension [105] as Figure 2.7 illustrates.

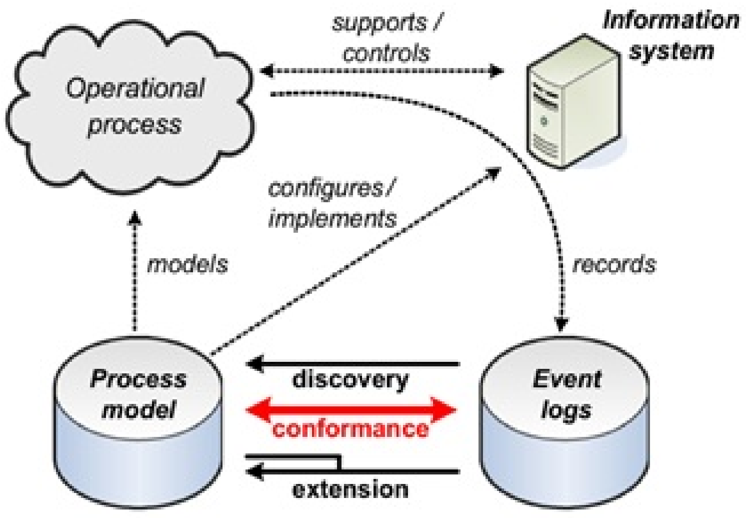


Figure 2.7: Process Mining Perspectives [105]

Discovery techniques are based on information mining from the event log data to present the control flow and dependency relations among tasks without reference to predefined models. Extension techniques support business process improvement based on discovered knowledge from the event logs. The interest of this study is in conformance techniques that support verification between event logs and prescribed models. The various forms of verification target different properties of the models. In the following section, a brief discussion of such properties is presented.

2.10.2 Business Process Verification Properties

Properties describe the normal expected behaviour of a model as prescribed during design. As discussed in section 2.8, model behaviour is exhibited following process perspectives [10] and verified according to the following properties. The various forms of verification target different properties of the models. In the following section, a brief discussion of such properties is presented.

***Soundness*:**

The notion of soundness property describes structural correctness of a workflow net [10]. The correctness of a workflow is determined by the fulfilment of the syntactical requirement of having each place or transition on a direct path from start to end. For a business process model to be sound it must have all its states as reachable, no deadlocks or live-locks, and should be able to terminate. There are additional correctness properties related to soundness, e.g. liveness, safety, Coverability and reachability. A summarised description is provided here under but details in [111] and [149].

- *Liveness* - verifies process models based on assumption that a process will execute successfully i.e. only good things will happen.
- *Safety* - implies that a business process model will execute successfully, and no bad thing is expected to happen.
- *Reachability* - each state in the model can be reached.

As a limitation to soundness as a verification property, only control flow perspective is checked and no or less regard to other perspectives of resources, data and time. Moreover, known application has addressed verification of traditional single party business processes, not much work has been realised from the point of view of multi-party collaborative business processes.

***Compliance*:**

Compliance is a property checked over models to verify their adherence to design requirements. Examples of such requirements are policies laws and regulations. In collaborative environments, compliance verification addresses variability checking i.e. verifying that model variants are true members of a given business process family [63, 61]. Compliance checking is as well been applied in change propagation verification especially where modifications are made from one end of the business partner or from the regulatory side [48, 49]. Models are checked to ensure that they remain compliant. In this thesis, attention is focused on supporting compliancy verification. The rest of the chapter presents related work in the area of compliance verification. We consider the contribution from their work, limitations and how this thesis addresses some of the concerns. To accomplish this target, a discussion of the state of the art in compliance verification is presented in the next section.

2.11 State of the Art: Compliance Verification Approaches

To support automated compliance verification, formal techniques, frameworks, methods and tools are applied to both traditional business processes and collaborative business processes. This section presents a cross section of the process model verification approaches selected on the basis of commonality, wide application and relevancy to the study. The presentation follows a brief description of the approaches, categorised based on formalism upon which they are based. However, a preamble is provided on the types of formalism.

2.11.1 Formalisms

A formalism describes a standardised known method or technique on which the approaches are based. A detailed description of formalisms is presented in [87].

Petri nets

Petri nets are based on Net theory. They describe means to provide descriptive, deductive and conceptual devices. According to Petri [120], petri nets offers 3 purposes;

"Descriptive devices for demonstrating the structure of systems and of processes supported by a system, in terms of axiomatically introduced concepts. Deductive devices for solving application problems such as; synchronisation problems, concurrency problems, problems involving mutual exclusion, conflict, arbitration, sequentialization, safety, problems of deadlock avoidance and of endless loop avoidance, problems in asynchronous switching logic, and last but not least problems

arising in an area not generally known as yet, called formal pragmatics in which we are concerned with the questions of the form 'What, precisely, do we do?', as opposed to formal semantics in which we are concerned with questions of the form 'What, precisely, does it mean?'; Conceptual devices producing precise concepts on many levels or for promoting the communication between the computer expert and other people;. . . "

Petri nets are adopted into workflow modelling to create workflow nets. Due simplicity, ease of access and formal mathematical foundation [1, 10], petri nets are a formalism on which several approaches and tools are based which specify and verify process models. A petri net is a bipartite directed graph with two types of nodes; Place represented as a circle, and Transition represented by a rectangle. Directed arcs connect places to transitions. Figure 2.8 illustrates a petri net based process model. Places contain tokens indicated using black dots while transitions have input and output places. Firing rules dictate how tokens move from one place to another [91, 118].

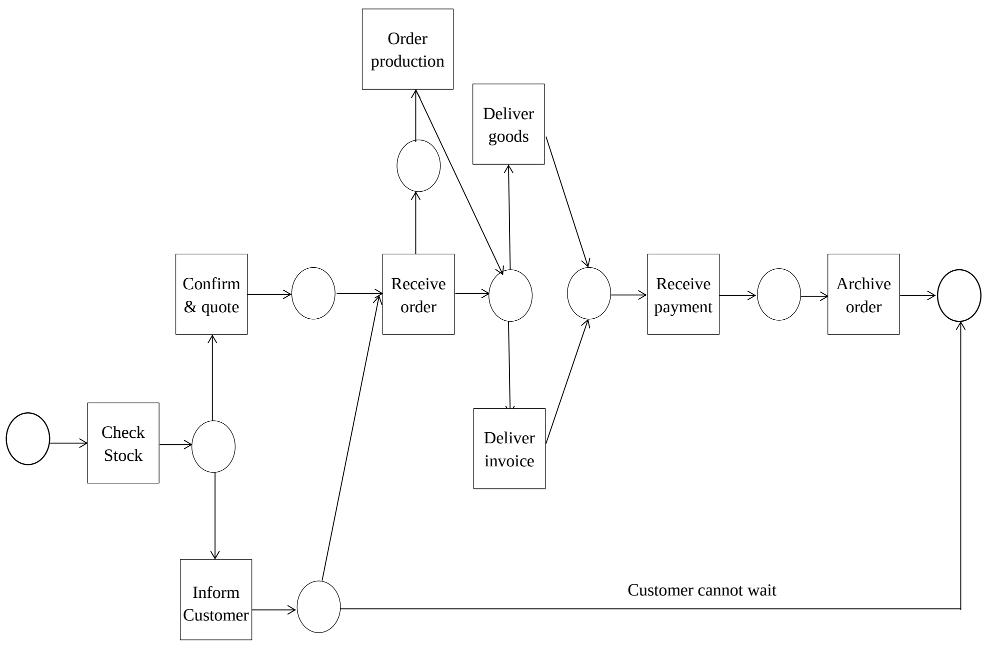


Figure 2.8: Illustration of a Petri net business process model

Business Process Modelling Notation (BPMN)

BPMN is a modelling standard from Object Management Group with a wide application because of its ease of use and intuitive graphical objects. BPMN graphical notations are easy to comprehend and apply for non-technical end users, analysts (responsible for creating the initial processes drafts), process developers (responsible for implementing the technology to perform the processes), business process managers and monitors [160]. Figure 2.9 is a simple order processing model illustrating use of BPMN notations. A customer creates an order via the seller's online system (e.g. via app) which is received and processed. The goods can only be shipped after the customer has processed the payment which is also confirmed by the seller. BPMN limitations include; lack of standardised semantics, and lacks expressiveness to support model verification.

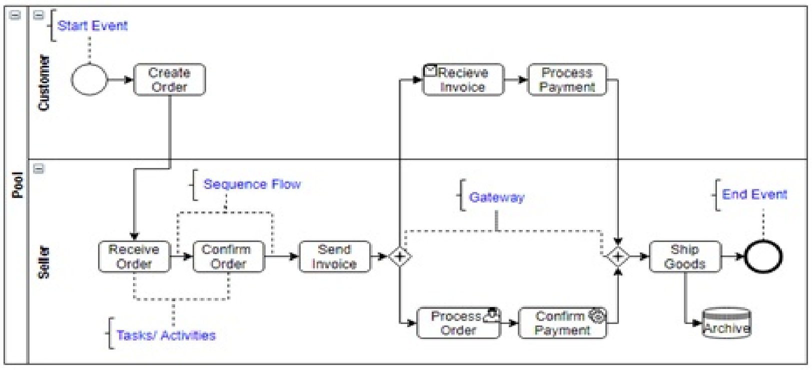


Figure 2.9: Business process Model showing BPMN Notations

Temporal Logic

Temporal logic is a formal method founded on mathematics. Models are specified and checked for correctness against a set of properties expressed as event orderings in time[97]. A set of temporal operators are used; Eventually $(\diamond, F)$, next-time (o), Always $()$, and Until (U). Temporal logic has two branches i.e. Linear Time Logic and Branching Time Logic. Model behaviour is specified while constraints and rules are expressed as logic formulae and verified against each other for conformity. Wide application of TL is known in concurrent, distributed, context aware and collaborative systems. With intention to support logical reasoning, different tools are grounded on different forms of logic like

Computation Tree Logic (CTL), Proposition Tree Logic and Timed Temporal logic [53, 18, 124, 69, 125]. However, temporal logic based tools are challenged by the state space explosion problem. To remedy the challenge model abstraction techniques are applied [51]. Another drawback of TL is the mathematical complexity associated with its use and application especially for non-expert end users.

Logic	Linear time (path-based)	Branching time (state-based)	Real time Requirements (continuous-time domain)
LTL	✓		
CTL		✓	
Timed LTL		✓	✓

Table 2.2: Classification of the temporal logics

2.11.2 Compliance Verification Approaches

in this subsection, a description of a set of existing verification tools and approaches is profiled.

Yet Another Workflow Language (YAWL):

YAWL is based on Petri nets and workflow patterns [8, 6, 7] to support modelling and verification by supporting early time detection of model errors. YAWL checks model correctness based on soundness property. Reduction rules and abstraction techniques are used to overcome state explosion [111, 145]. Modifications to enhance verification capacity were implemented to cater for cancellation regions and OR Joins [163, 153].

Workflow Analyser (Woflan):

Woflan is an independent model verification tool with capacity to give spot on diagnostic information to repair detected errors [152]. It integrates with WFMS like COSA, Staffware and Protos to verify models for structural and behavioural properties of soundness [1, 149, 152] .

Coloured Petri Nets- CPN Tools:

The set of tools are used to specify and verify models for reachability, Liveness and boundedness by employing state space methods and model checking. It considers time to execute activities in the system and support simulation, performance analysis and verification of models for soundness through computation of reachable states and state changes of the model represented as directed graphs [81].

Declare:

An approach to design and verify flexible and dynamic process models using constraints based approach. As a declarative language, models are specified by stating rules to be followed and support their verification. Dead activities, conflicting constraints and changes in models are verified based on control flow perspective. Declare also supports design of declarative modelling languages including ConDec and DecSerFlow [117, 118, 119].

Challenges of Petri net Based Approaches

The above approaches are based on Petri net formalism to specify, model and verify business process models. Their application has mainly been characterised by the following challenges; they profoundly support verification of control flow constraints, no known application to collaborative business process model verification, their capacity to support dynamic constraints checking is limited. Moreover, the approaches focus on detective approach to compliance verification using process mining technique, i.e. given a log and some property, check whether the property holds [4]. These challenges call for an approach that can leverage the limitations and capacity of YAWL, Woflan and CPN tools as petri net based approaches. In the next subsection, a discussion of temporal logic tools follows.

HYbrid TECHnology (HyTech):

A logic-based approach employing symbolic computation for automated verification of models against properties specified in real time temporal logic. Symbolic computation is a procedure for verifying ICTL formula over hybrid automata [14] where as a hybrid system is one composed of a discrete program embedded within a continuously changing environment and interacts with the environment in real time [71]. E.g. Hybrid automata is a generalised finite state machine with both discrete and continuous variables. HyTECH is applicable for verification of mission critical systems for reachability, Liveness, time boundedness and duration. HyTECH+ [70] which is an extension to the classical HyTECH. Symbolic Model Verifier (SMV): SMV uses binary decision diagrams to check models where states and transitions are in a single block than a single state at a time [38]. NuSMV a later version expresses specifications in LTL and CTL to analyse and verify synchronous finite-state and infinite-state systems for correctness, liveliness, and safety [37, 85].

UPPAAL:

The approach employs automata logic is applied for real time simulation and check on system behaviour for reachability, invariability, safety, non-zenoness and bounded response in real time systems [94, 95, 21]. Diagnostic trace for property violation is generated.

SPIN:

Verifies asynchronous system specifications expressed in PROMELA against properties specified as temporal logic formula and converted into Buchi automatons to compute the product of the claims and the automaton representing the global state space. The resulting automaton is then checked, if it is empty it implies the claim is not satisfied otherwise it contains the behaviour that satisfies the original formula. Partial order reduction method is employed to control state explosion [120, 76, 77, 75]. Safety and Liveness properties are checked.

The LTL Checker

LTL Checker is an event log-based tool for verifying conformance of a process to a set of properties. Ordering and timing properties are formulated from the logs and checked against the business process model to verify whether they hold or not [4, 20]. The tool is implemented as a plugin into the ProM framework. Medeiros et al. propose semantic process mining as a way to add meaning to labels and data used as the basis of log mining to construct models and provide adaptable and reusable solutions understandable for process analysts [105]. The LTL Checker is thus extended to perform semantic event log mining.

TLA+:

Temporal logic for action is a concurrent system abstraction tool that supports model specification and checking. It enables writing of algorithms, translating them into system specification models and checking them for correctness against deadlocks, termination and invariants [11, 93, 158].

Business Process Variability Tool (BVP):

A declarative tool supporting design time preventative specification and verification of business process model variants for conformance with the reference process. A reference process is the core process from which variants are created through configurations and customisation to meet specific requirements of different partners in the process. Using basic principles, a business process template is created that is then used to create and validate process variants. The reference process is expressed as a formal specification while process variants as a system model; conformance is formally verified through model checking [60, 62].

Compliance Request Language (CRL):

The language is built as part of a compliance management framework to simplify and support abstract pattern-based design time preventative compliance requirements definition and enforcement on business processes. The language is grounded on temporal logic utilising formal reasoning to support compliance checking. The patterns formalise the compliance requirements covering the traditional business process perspectives of control flow, resources and temporal perspectives. Besides, compensation and monotonic requirements are as well supported [52, 46]. To verify the requirements, the tool relies on SPIN model checking tool which implies that the outcome is affected by state explosion problem. Even then, conflicts and inconsistencies may exist in the requirements that may bring the process to a deadlock [156, 157]. These must be verified for non-existence

Deontic Logic Languages

Deontic logic is a formal system used to specify obligations, permissions and responsibilities among contractual parties. The logic is well known for supporting formalisation of contracts and verification for compliance of parties and stakeholders to the obligations, permissions and responsibilities. Deontic logic is the philosophical basis for PENELOPE and Formal Contract language.

Formal Contract Language (FCL) FCL supports normative specification and verification of process models for compliance against obligation and permission constraints. The normaliser and the inference engine facilitate reasoning over normative rules. The normaliser explicitly derives and merges rules to their normative conclusions to remove redundancy and identify conflicts while the inference engine derives conclusions from some input propositions. It also embeds defeasible logic a non-monotonic formalism for constructive proofs to allow trace of derivation upon conclusions substantiating violations [131, 130].

PENELOPE Language – Process Entailment from Elicitation of Obligations and Permissions: The approach is based on the notion of explicit definition and expression of business policies and regulations as constraints imposed on business processes in a declarative way at design time. Control flow perspective is emphasized i.e. the sequence and timing of process events. The language is implemented as a supporting framework to generate business process models compliant with policies and regulations i.e. the obligations and permissions. The obligations and permissions are expressed using deontic logic and used to generate a non-executable control flow based compliant process model applicable for verification and validation of violations in other models for properties like deadlocks, livelocks, deontic conflicts, temporal conflicts, and trust conflicts [56, 57, 55].

The logic based approaches presented in the above section have as well faced limited application due to a set of challenges; state space explosion problem is a big challenge that many model checking tools face. However, the application of abstraction methods and techniques [51] mitigates the challenge. Other draw backs involve the associated complexity for non-expert end users without competency in mathematical skills. At this point, we the second requirement to be satisfied by our approach is derived.

- ***Req2. Compose a less complex mechanism (language) to support non-expert end users to define and specify compliance constraints based on general policies and regulations.***

BPMN-Q:

BPMN-Q is a compliance checking language built as an extension to BPMN to query models segments where modifications have occurred, and verify their compliance with ordering constraints of activities. Process designers can query business process models based on their structure to derive patterns, variants as well as compliance to quality constraints specified by international standards and regulations like ISO and total quality management. The authors further extend the language to cater for data and temporal constraints. BPMN-Q however is a detective compliance monitory approach for supporting adherence queries as opposed to the preventive approach [15, 17, 144].

SecBPMN-Q and SecBPMN:

Salnitri et al. extend BPMN into SecBPMN to support the definition and specification of information systems, while BPMN-Q is extended to SecBPMN-Q to supprot expression of security policies, requirements and constraints. The specifications are then verified via an engine [132]. As a security policy specification approach, checking other forms of constraints is not implemented.

Protos:

An approach that specifies models and supports verification through simulation based analysis based on data, user, or control flow perspectives [58].

SeaFlows Toolset:

SeaFlow is a framework tool for compliance verification with support for model abstraction. Structural compliance verification derived from compliance rules is supported at design time as well monitoring at runtime. While, behavioural compliance checking caters for data constraints Compliance rule graphs are used to model compliance rules [107, 98, 122, 92].

Conformance checker

The approach checks for conformance and fitness between the process model and specified behaviour i.e. the extent to which the log traces can be associated with valid execution paths specified by the process model, and appropriateness i.e. the degree of accuracy in which the process model describes the observed behaviour [126]. The tool is a detective approach and non-applicable for collaborative prevent scenarios.

As the discussion has revealed, a plethora of verification frameworks, methods, techniques and tools are in existence. A summary of these approaches is presented in Table 2.3.

2.12 Verification Requirements for Collaborative Business Process

Unlike traditional business processes, Collaborative business processes span beyond borders of a single organisation. Due to their nature, Collaborative Business Processes present unique characteristics and verification requirements that cannot be automatically met by traditional business processes verification approaches. It necessitates consideration of other factors to achieve with their verification;

- Need meet data requirements sourced from different partners to the collaboration.

- Need to express interactivity and communication requirements among the collaborative business process eco-system. In most cases, dedicated communication and interaction protocols are established for message exchanges between the partners to engage in discussions and iterations before reaching a decision [86]. Due to different role interactions from partner organisations, it is important to clearly define organisational units, communication channels and reporting relations during executions [171]. The verification of organisational units, communication channels and reporting relations prevents role conflicts and makes work swift. A platform independent model for specifying cooperation among workflows is proposed by [34]. A system implementing the model is as well presented allowing workflows to publish and subscribe to events and, definition of points in execution where to send and receive events. The events are filtered, correlated and dispatched to appropriate target workflow instances. The model however does not guarantee safety of interactions or cooperation among workflow systems since no verification is conducted. Common forms of interaction adopted in collaborative business processes include [9];
  - Capacity sharing; resources are distributed but under one managerial control
  - Chained execution; the process is broken into sub processes that are executed by different partners
  - Subcontracting; phases of the process are sub contracted to other business partners
  - Case transfer; work is balanced between partners
  - Loosely coupled; the process is partitioned horizontally and each partner runs one or more parts of the process over a defined protocol
- Dynamism, Flexibility and Complexity: Complexity results from various requirements from the stakeholders. These usually must be satisfied by the process to achieve compliance on top of other requirements from the external agencies. This requires high flexibility and dynamism of the process. The rate and speed at which changes are verified, integrated and propagated to the necessary components is

important to facilitate decision making. Short of this would be detrimental to the collaboration and entire supply chain.

- Security and privacy requirements: Collaborating organisations do not entirely share their workflows but only necessary data or part of the workflow are made visible. Organisations desire to retain their autonomy [27]. This implies defining the scope of the collaboration sphere and supporting interfaces, and verifying that privacy is not compromised. For example, a Virtual enterprise coordinator (VEC) is proposed as an approach to control and maintain privacy, flexibility and independence of an organisation participating in a Cross organisational business process. VEC supports mapping between interacting workflow management systems by defining interfaces that preserve privacy of shared workflows. The approach however disregards set up and distribution of agreements between collaborators and does not verify the implementation.

- Semantic Notation Requirements: Collaborative business process models are often composed bu merging existing partner models into a single model. This way, model semantic ambiguity arises due to the different ways in which each partner has been designing their models or the supporting modelling languages used at each end. Some modelling tools lack uniform semantics, e.g. BPMN tools. Unifying the semantics to avoid misunderstandings is a requirement whose satisfaction is achieved through semantic annotation verification.

Table 2.3: A summary of general process verification approaches

Approach	Properties	Flexibility	Arbitrariness	Suitability	Complexity	Limitations
Woflan	Soundness and Liveness	Verifies completed models	Single model verified at a time	Specific to models developed in particular language	Easy to use with graphical interface	Non-collaborative Complex outcomes It is difficult to trace errors
YAWL	Soundness and Liveness	Design time exception handling model	Each model or sub model is verified independently	Verifies control flow based on Resetnets and transition invariants	Not complex to use and supports extension plugins Graphical interface	Non-collaborative
FlowMake	Structural conflicts like synchronisation, Deadlocks, consistency, Liveness	Exception handling. Not scalable as models grow large	Not domain specific Sub models are verified independent of main model	Lack data perspective which is very essential for vF cBP	Graphical interface makes it usable for non-expert users	Non-collaborative Based on control flow and abstracts other perspectives It is difficult to trace errors

Coloured Petri Nets	Performance analysis Coverability and occurrence	Supports exception handling	No support for verification of main and sub models	Concurrent systems Non domain specific	Graphical tool with less complexity	Non-collaborative support
SPIN	Correctness and logical consistency	On Timeouts it supports exception handling	Wide application not limited to particular domain	The richness of temporal logic can make it viable for vF cBP	Its syntactical structure and semantics make it complex. With XSPIN a graphical interface is provided	Non-collaborative State explosion Restricted to smaller systems
UPPAAL	Bounded Liveness, deadlock freeness and deadlines	on-the-fly verification but not scalable	Verifies concurrent systems but not simultaneously.	No support for data analytics	Supports diagnostic trace leading to source of errors	Non-collaborative support
KRONOS	Reachability properties (Safety, Non zenoness, Bounded response	Exception handling supported	No simultaneous verification models and sub models	No known application to vF domain	Graphical interface usability Provides counter examples to aid verification	Non-collaborative Limited to smaller models Co consideration for data

SMV/ NuSMV	Correctness, safety, and liveliness	Support for exception handling	No support for simultaneous models or sub model verification	Non domain specific Scales to other applications	NuSMV - graphical interface to ease usability Counter examples provided	Non-collaborative State explosion
HyTECH	Reachability, Safety, Liveness, time-bounded, duration	No exception handling. Not scalable	Specific application for embedded and hybrid systems	Lacks elements like data which a key to vF cBP	Complex tool due to its syntactical and semantic requirements	Non-collaborative State explosion Supports to smaller systems
Woflan	Soundness, Liveness and Reachability	Lack of flexibility	Single model verified at a time	Verifies models from other languages	Graphical interface for usability	Non collaborative. Output not easily understandable
ADEPT	Semantic correctness, deadlock and Safety	Handle exceptions Flexible verification	No support for simultaneous model and sub models verification	Applicable to other domains other than clinical processes	Use of process templates to easily create processes.	Lack of proven application.

2.13 Summary of Requirements of Research and/or Exciting Approaches for Compliance Verification

As the discussions have shown, verification of collaborative business processes relates to various aspects categorised under control flow, resource flow, data and temporal groupings. We have noted that verifying for compliance of collaborative business processes scales beyond the requirements for verifying traditional business processes due to the unique characteristics of these processes. Further more, different approaches have been presented that target compliance verification. However, these are limited in several ways as indicated in Table 2.3. In [88], we justified the need to verify collaborative business processes for compliance. Moreover, in section 2.2 policies and regulations were described and their examples cited in section 2.6. The policies and regulations form sources of compliance requirements and constraints to be satisfied by the business processes. Based on the discussion and analysis of related work, the gaps identified formed propositions to the requirements necessary for an integrated compliance verification approach for collaborative business processes. The gaps and requirements are summarised below:

- Limitation 1: Existing approaches and related work are based on either a single constraint category or a combination of two categories. Profoundly, control flow is well addressed as seen in work by [15, 16, 119, 117] shows. Others have addressed resource constraints as a single category, e.g. [112, 30, 31] Ortega et al. 2013. While others have addressed data specific categories [16] and [92], and temporal specific approach [129]. In other cases, a combination of categories is addressed, e.g. in [17, 98, 60] Awad et al address compliance of process changes in terms of control flow and temporal requirements, while Ly et al address process compliance in terms of control flow and data requirements whereas Groefsema proposes a tool for variability compliance checking based on control flow and data requirements. Compliance to data and resource requirements is addressed by [122]. However more related work attempts are made to address control flow, resource, data and temporal constraint categories in single approach. For Example [52] implements a compliance request language for specification and definition of constraints while [142] all the constraint categories are used to illustrate how to achieve compliance. However, the all categories based compliance checking is not achieved from the composed frameworks but outsourced to compliance checkers. for example in [52], the composed CRL language is only applicable to in the definition and specification of compliance requirements.

Requirement 1: Compose a constraint based compliance verification approach supporting multi-level compliance checking.

- Limitation 2: In addition to the above, the cited related work addresses either design time, runtime or post execution compliance checking strategies. The design time based approaches are limited to only data available at design time and ignore checking scenarios that depend on running data. Runtime approaches and post execution approaches are known to take place late after a damage resulting from non compliance has occurred. For example resource based compliance approaches in [80, 113, 96] are based on process mining of event logs to support resource allocation [96] or to understand resource behaviour [113]. Whereas the design time approaches like the one presented in [119, 117] are not exhaustive to consider runtime data. Work from [52] and [142] cuts across strategies. However, the limitation in this work has been discussed under limitation 1 above.

  Requirement 2: To compose a hybrid compliance approach whose verification and checking application is not limited to either design time, runtime or post execution but supports all strategies. category of constraints.

- Limitation 3: Complexity associated with related approaches resulting from the inherent supporting logic on which they are based. For example CRL [46] is based on a combination of logic involving LTL, Metrical Temporal Logic (MTL) and ForSpec Temporal Logic. Work by Awad also employs LTL and past temporal logic [15] while Goedertier and Vanthienen apply deotic logic to model and verify permissions and obligations [56]. Despite the justification given by the authors, the logic remains complex for ordinary users necessitating for a less complex mechanism that subject matter experts who are not modelling experts can use to formulate, specify and verify process models.

Requirement 3: Compose a less complex compliance checking mechanism to support non-expert end users.

- Limitation 4: Platform independence. Besides Woflan which integrates models from other workflow systems like Staffware, COSA etc most of the approaches discussed above that translate into software tools are not platform independent. This implies that the modelling and verification can only be done with that specific tool resulting into model lock-in. A platform independent verification tool enhances its portability and makes verification work flexible and swift without tying users down to a specific tool. It is also easy to maintain and update requiring less time and cost.

  Requirement 4: Compose a verification environment that is platform independent. One that accommodates and verifies system models regardless of their design or modelling environments. The implementation architecture is based on the service oriented architecture to achieve a service based hybrid verification approach which is not limited to a single strategy of design time, runtime or post execution.

2.14 Summary of Related Work

This section summarises the related work presented in sections above. The related work shows compliance and BPM as two related fields who research is vast and still growing in differing dimensions. The chapter discussed the concept of policies and regulations as a basis for compliance requirements and constraints. in addition, the structural facets of the business process were discussed as a means to study process behaviour and their importance to provide a taxonomy under which the compliance constraints were categorised in relation to constraint patterns. These groupings include control flow patterns, resource constraint patterns, data constraint patterns and constraint patterns.

Further to the above, process verification was discussed under which the state of the art was presented in lieu of collaborative business process verification. The outcome at this level is the comparison assessment of compliance verification frameworks, tools and techniques presented in section A.1 of the Appendix. From the analysis, gaps and limitations were identified which formed a basis upon which the requirements for our proposed collaborative business process compliance verification.

Chapter 3

Methodology

3.1 Introduction

This chapter describes the methodology used to achieve the outcome of this report. Guided by the research question and objectives stated in the chapter one, this chapter presents the procedural steps, research approach and tools used to accomplish the research. The chapter is structured as follows: Section 3.2 presents the conceptual framework upon which the subject foundation is based. Section 3.3 is concerned with the research approach applied in the study. An inductive approach was followed in this case. Section 3.4.1 presents the research methodology while section 3.4 presents the research design. In section 3.4.2 where, Design science method is introduced and described. In Section 3.5 We show how design science was applied to achieve the artifacts of the research by following the recommended steps presented in section 3.6. The chapter concludes with summary in section 3.7.

3.2 Conceptual Framework

The conceptual framework presented in Figure 3.1 provides a conceptual foundation upon which the concepts in the study are derived. Between business process management and compliance management disciplines lies the compliance gap. The gap can be bridged by having business processes that comply with policies and regulations. Achieving compliant business processes creates the need compliance verification mechanisms, methods or techniques that facilitate checking conformity of the business process with regulatory requirements. Analysis of the gap, concepts and requirements from the two disciplines enabled formulation of the research question, research scope, objectives and aims of the research.

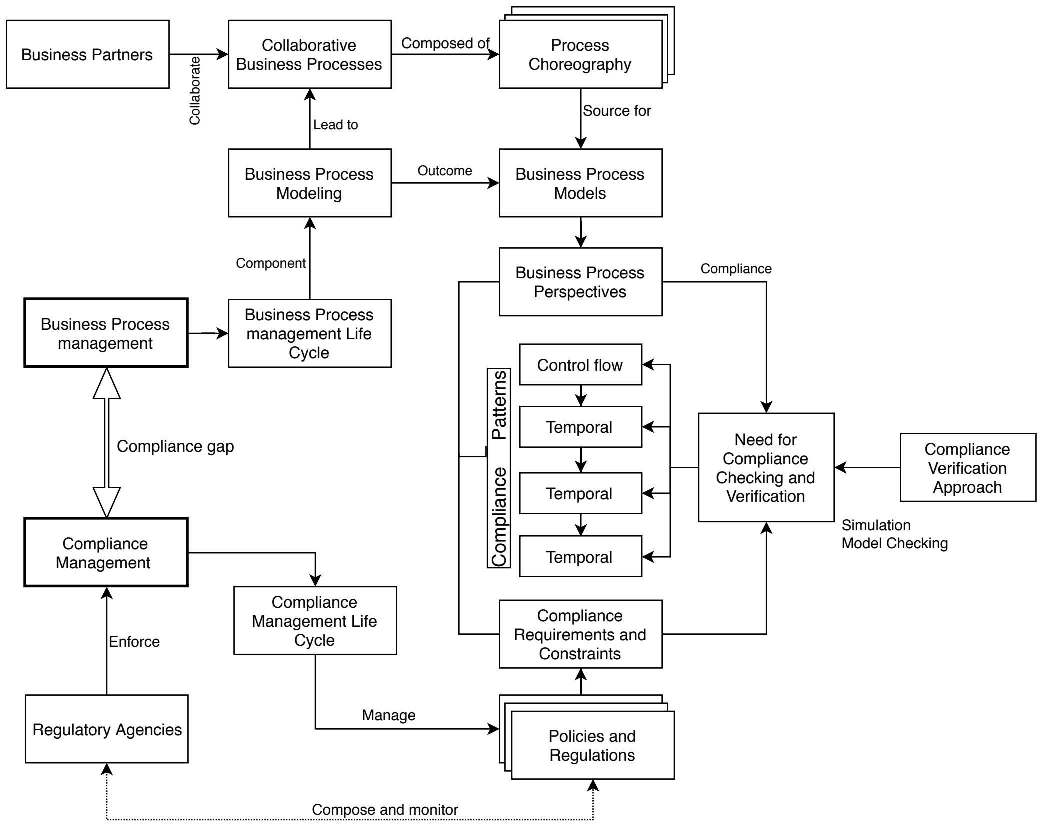


Figure 3.1: Conceptual Framework

Besides the major concepts, a brief description of other concepts is provided [28]:

- *Business Process Management (BPM)*; a holistic management methodology to deliver value to customers and stakeholders by managing efficient and effective business processes.

- *Business Process Modelling*; a graphical representation of processes to enable analysis and improvement of the current process.

- *Business Process Modelling Tool*; a software tool to create business process models. Process Model; a graphical representation of a business process that exhibits the activities and their inter-dependencies that make up the business process to any desired level of detail.

- *Capacity Constraint Resources*; where a series of non-bottlenecks, based on the sequence in which they perform their jobs can act as a constraint.
- *Constraint*; a condition restricting or regulating a process and usually constraints are outside the control of the project team. Failure to meet a constraint may causes an exception condition or other defined procedure.
- *Compliance*; an act of adhering to any standards, procedures, or processes established as necessary for operational effectiveness.

3.3 Research Approach

A research approach describes a planned procedure involving a set of steps detailing application of methods to collect, analyse and interpret data. Two broad categories of research approach are deductive and inductive approaches. The distinction between the two approaches lies in the fact that the deductive approach tests for validity of hypotheses or theories whereas the inductive approach works towards emergence of new theories and generalisations based research questions, goals and objectives [29]. In this

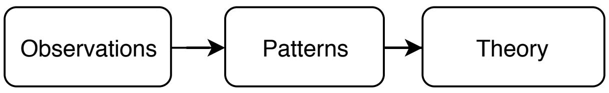


Figure 3.2: An Inductive Research Approach [29]

thesis an inductive research approach was adopted to study policies and regulations, compliance requirements and constraints as well as collaborative business process requirements. These were generalised and categorised into known constraint patterns, formalised and represent as model logic upon which reasoning is applied. The reasoning outcome informs whether the process is compliant or non-compliant. A compliance verification approach composed of verification algorithms is designed and presented as an artifact to support verification of collaborative business processes for compliance. Procedurally, the approach involved the steps as follows

- Systematic analysis of both regulatory requirements and collaborative business process verification requirements formed the initial step. The step provided an understanding of the requirements, expectations and compliance concepts in explicit terms to support verification.

- Furthermore, a less complex mechanism to formalise requirements into compliance constraints according to pattern categories. Both description logic and temporal logic. The outcome are formalised constraints.

- Verification algorithms are designed based on categories of constraints. These are applicable in verification to check for existence of required behaviour in the business processes.

- Lastly, the verification algorithms are evaluated using industry based use cases. Furthermore, architectures for practical implementation and evaluation of artifacts are presented.

3.4 Research Design

A research design forms the basis upon which the research plan is drawn. In this thesis, the research problem was contextualised by breaking it down into different components according to the two discipline studied (see Figure 3.1). The conceptual framework informed the research approach. The research was compelled by some known compliance related challenges as discovered from literature (chapter 2) and in practice that have posed a knowledge gap worthy a research. Despite existence of probable solutions in form of artefacts and tools from both industry and academia, some of the unresolved identified gaps are addressed by the artifact we present as a solution to the research problem defined in chapter 1. With consideration of other research methods like grounded theory [141, 54], design science [74] was deemed most appropriate to achieve the outcome of the study.

3.4.1 Research Methodology

Various research methodologies available for information systems (IS) research. For instance, Grounded theory (GT) [29] which leads to discovery of theories from data systematically obtained from research. Besides its application in various IS research projects to study different phenomenon, various studies have specifically promoted the use of GT in IS research. For example, Martins et al, demonstrate the application and use of grounded theory in three different IS research projects and share lessons learned from its use to provide well formed views about its use in IS research projects [102]. In another study [103], the use of GT in IS research is promoted by investigating alternative approaches. A clarification of the nature of grounded theory approaches in terms of epistemology stance as a positivist or interpretivist is provided while recommending its use in combination with other theories due to its multi disciplinary nature.

However, design science [74] was adopted as a suitable method to accomplish the goals of this research with justification given in section 3.4.2.

The information systems field deals with artifacts in an ecosystem environment where socio-technical systems operate. Socio-technical systems (STS) express the interaction between people and technology (e.g. computerised information systems) in organisations and further recognise the interaction between society's complex structures and human behaviour. Besides the behavioural aspect of the STS, design science also considers STS from an engineering view making it suitable as a method for engineering leaning research projects [73]. The choice and application of a methodology depends on its usefulness in achieving the objectives and expected outcomes of the research project. Therefore choosing design science was based on the aims, objectives and expected output of this study.

3.4.2 Design Science

Design science extends the human and organisation abilities to create and evaluate new and innovative artifacts. The artifacts are in terms of constructs, models, methods and instantiations created as solutions to existing problems in the community based on knowledge and understanding of the problem domain [72] in a socio-technical environment.

Constructs provide a framework within which problems and solutions can be defined and communicated, e.g. a language, model or method. Models represent the reality e.g. a world phenomenon used to support design of the problem and its solution space. Methods represent mechanisms used to define solutions to problems, e.g. formal formulae, mathematical algorithms, or informal textual descriptions like best practice approaches. Instantiations facilitate the implementation of constructs, models, or methods as a working system and assessment of their feasibility and suitability to its intended purpose.

Design science overlays research in 3 interrelated cycles of Relevance, Design and Rigour (3.3). The relevancy cycle seeks to improve the environment which is the problem space, by providing solutions relevant to existing problems in form of artifacts. The artifacts are returned to the environment for evaluation. Design science recommends rigorous testing of the artifact before its release. This way, multiple iterations take place before the artifact goes to the relevance and rigour cycles. The rigour cycle refers to the application of knowledge from well-known grounded scientific theories and engineering methods. Knowledge may come from experiences and expertise that define state of the art or existing artifact and processes. The relation to the knowledge base ensures a new and innovative artifact is presented as a contribution. The design cycle promotes the design and evaluation of the artifact. The evaluation provides feedback to fine tune the artifact to its final state of application [74]. Figure 3.3 illustrates the design cycles. In

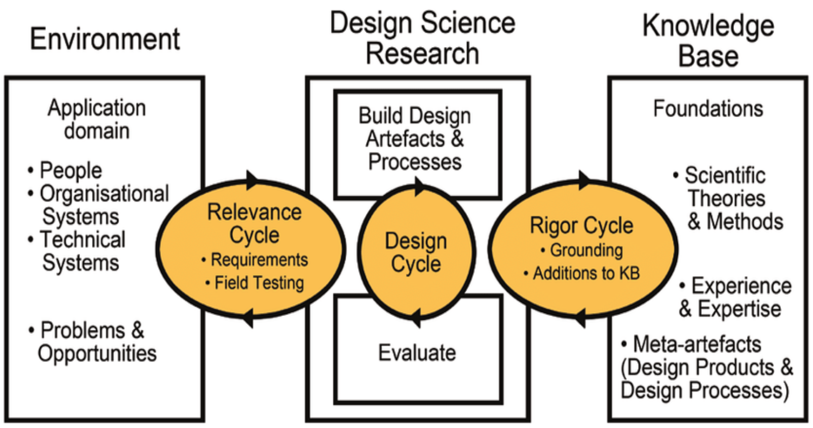


Figure 3.3: Design Science Research Cycles [74]

specific iterative terms, we follow the described steps of design science (Figure 3.4 as described by [116]. As stated above, the problem definition of this research is motivated by the need for a less complex compliance verification approach in practice, the objectives of the intended solution are clearly defined as the initial steps. Afterwards the verification algorithms are designed, demonstrated, evaluated and results communicated.

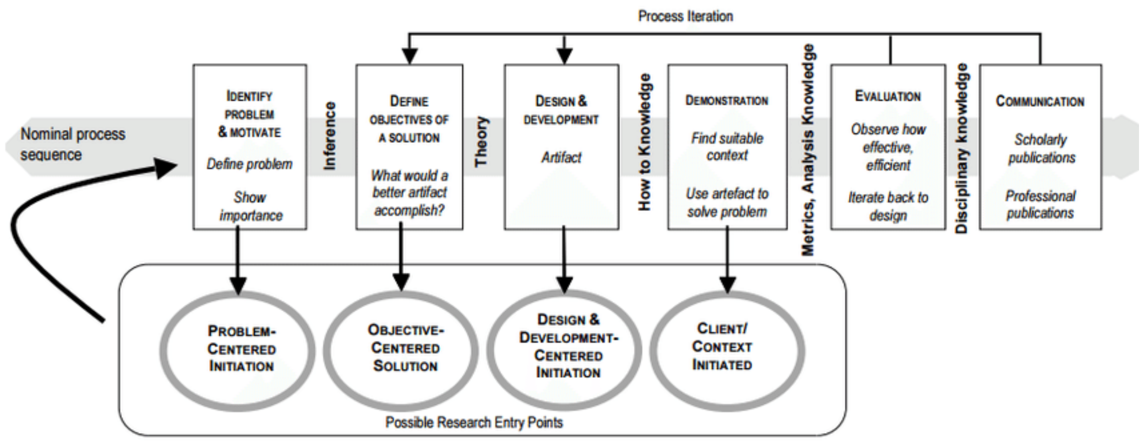

Process Iteration
Nominal process sequence
IDENTIFY PROBLEM & MOTIVATE
Define problem
Show importance
Inference
DEFINE OBJECTIVES OF A SOLUTION
What would a better artifact accomplish?
Theory
DESIGN & DEVELOPMENT
Artifact
How to Knowledge
DEMONSTRATION
Find suitable context
Use artefact to solve problem
Metrics, Analysis Knowledge
EVALUATION
Observe how effective, efficient
Iterate back to design
Disciplinary knowledge
COMMUNICATION
Scholarly publications
Professional publications
PROBLEM-CENTERED INITIATION
OBJECTIVE-CENTERED SOLUTION
DESIGN & DEVELOPMENT-CENTERED INITIATION
CLIENT/CONTEXT INITIATED
Possible Research Entry Points

Figure 3.4: Design Science Research Methodology (DSRM) Process Model [116]

3.5 Application of Design Science in BPM Research

BPM as a growing discipline creates contribution to the computer science community through the artifacts created. According to [100], the maturity and relevance of the BPM discipline is emphasised by reviewing and examining BPM work from different publications and recommend to progress BPM as a Design science by;

- Creating taxonomies to structure the field and relevant processes.
- Extend engineers techniques beyond process analysis and control flow perspective to consider other perspectives and roles.
- Make use of scientific research methods like case studies and action research as used in IS research.
- Develop explicit definition of hypothesis for the algorithms that BPM research usually produces. E.g. on the benefits intended to be achieved. Make benchmark data publicly available.

Following the recommendations, our research extends verification beyond control flow, makes use of industrial use cases for evaluation and validation of the artifact, and the designed algorithms are demonstrated to show their efficacy and applicability. The following section describes specific application of design science in this research.

3.6 Application of Design Science in our Case

Design Science was preferred over GT to achieve the outcomes of this thesis. As highlighted in previous sections, GT supports construction of new theories and knowledge based on collection and analysis of data. Such methodical requirement limited its application in this thesis where use cases were preferred as opposed to collection of data. Design science was appropriate due to the nature of the research i.e. the research was set to provide a solution to an existing problem space in the environment, and support the rigorous evaluation and the application of knowledge using well known theories. The following section illustrates how categorical procedure followed according to the design science cycles and the six research activities. Peffers et al [116] categorise IS research into six activities as a way to establish commonly acceptable framework for conducting IS research based design science principles. Figure 3.4 presents a summary of these activities which include:

The first step is to *identify the research problem* to be addressed by the research and its motivation.

Secondly, the *objective of the solution* is specified showing what the artifact would accomplish.

Thirdly *design and development* of the artifact follows, this is the actual implementation of the solution which addresses the problem.

Fourthly, the *suitability of the artifact to solve the problem* domain is demonstrated.

The fifth activity concerns *evaluation* of the artifact to establish how effectively and efficiently it solves the problem or how well it meets the design requirements.

Lastly, the sixth activity concerns the *research outcomes and how they are shared, communicated and publicised* to the community. This is achieved through scholarly and professional publications.

Based on these groupings, specific steps undertaken in the research process are illustrated with mapped Figure 3.5 showing the application of design science into our research process and compliance verification approach. The description of how each step was accomplished follows:

- *Problem identification and motivation*; the initial step towards the research presented in this thesis was to define and specify the research problem. The problem was motivated by the continuous compliance problems and challenges despite existence of several solutions. The solution targets non-expert end users like process modellers and compliance officers that need support to design and verify compliant collaborative business processes. Existing compliance management framework are not end user specific and do not target collaborative business processes. In this case the problem space included stakeholders like business organisations and regulatory agencies who specify collaborative business process and regulations respectively, process modellers and compliance officers.

- *Objectives of the solution*: The second step in the research process involved setting the research objectives. These were derived from the research question. The major research objective and sub objectives guided the study by providing a direction, research goals and scope towards the solution.

- *Design and development*: The third step involved solution design. To achieve the design, an artifact in form of a compliance verification approach was developed that includes among other components;

  (a) A compliance requirements. elicitation and definition mechanism

  (b) A mechanism to translate requirements into compliance constraints by formalisation using model logic for automated interpretation and reasoning.

  (c) Constraint specific compliance verification algorithms that detect constraints violations and deviations from desired behaviour.

- *Demonstration*: The fourth step involved solution demonstration. This way, Chapters 8 and 9 present the demonstration of constraints specification and verification algorithms based on two industry use cases. In the first case of demonstration in section 8.4, a consolidated form of verification was demonstrated to check the entire process for compliance violations using an overall algorithm 12 whereas in Chapter 9 constraint specific algorithms are demonstrated based on use case 2. In each case, the algorithms are used to reveal compliance status of the business process, whether compliant or otherwise. If non-compliant, the source of violation is reported.

- *Evaluation*: In the fifth step the algorithms are evaluated as Chapter 9 presents assessing their applicability and efficiency to verify process compliance. For evaluation, the second use case is specifically used. The evaluation is conducted in two perspectives;

  a. Using MEM model to evaluate the efficacy of the algorithms. The outcomes are reported in section 9.1.

  b. Performance evaluation, the algorithms are assessed in terms of their ability to detect violations and their performance capacity in terms of time and space requirements.

  The outcome of the evaluation showed the efficacy and applicability of the solution to solving the compliance verification problem. Moreover, the compliancy algorithms are expressive enough to meet the verification requirements for the use cases.

- *Communication*: The research has been communicated in both academic and professional circles; conference publications, EU H2020 FIRST project, knowledge application to the industry with GK software Inc. GK is a software company that deals with data for its clients and thus must have conformance of its business processes, products and services with data regulatory laws.

Figure 3.5 illustrates the application of the design science cycles (Relevance, Design and Rigor cycles) as well as the research activities to achieve the outcomes of our research. Based on the relevancy cycle, as well as the research activities identify problem and motivate, and define objectives of the solution, the problem of the study was defined and motivated. These are described in chapters one and two in which we defined the problem (chapter 1.11) and motivated with support from related work (chapter 2). Using the design cycle together and following design and development activity, the research artifact i.e. the compliance verification approach was designed based on inputs from chapters 4, 5 and 6. The artifact is presented in chapter 7. The demonstration and evaluation of the artifacts as further required by the research activity 4 and 5 was accomplished in chapters 8 and 9 using two industry based use cases. The major outcome of the thesis, the compliance verification approach together with the components i.e. the compliancy requirements elicitation and formalisation mechanism and the verification algorithms after rigorous evaluation and validation are useful contributions to the existing body of knowledge in business process management and compliance verification. The requirements of activity six i.e. communication of the research are accomplished through the presentations at workshops, conferences as well as publications in conference proceedings and journals.

3.7 Chapter Summary

The chapter presented the methodology followed to accomplish the goals and objectives of this research. The research process started with a conceptual framework which provided a technical foundation for the study's BPM discipline and then followed an inductive research approach. Design science was adopted based on analysis of other IS research methods like the ground theory as the justification revealed. The research steps that design science recommends were then followed to accomplish the study. As Figure 3.5 shows, these steps are mapped into our compliance verification approach indicating research outcomes for each chapter.

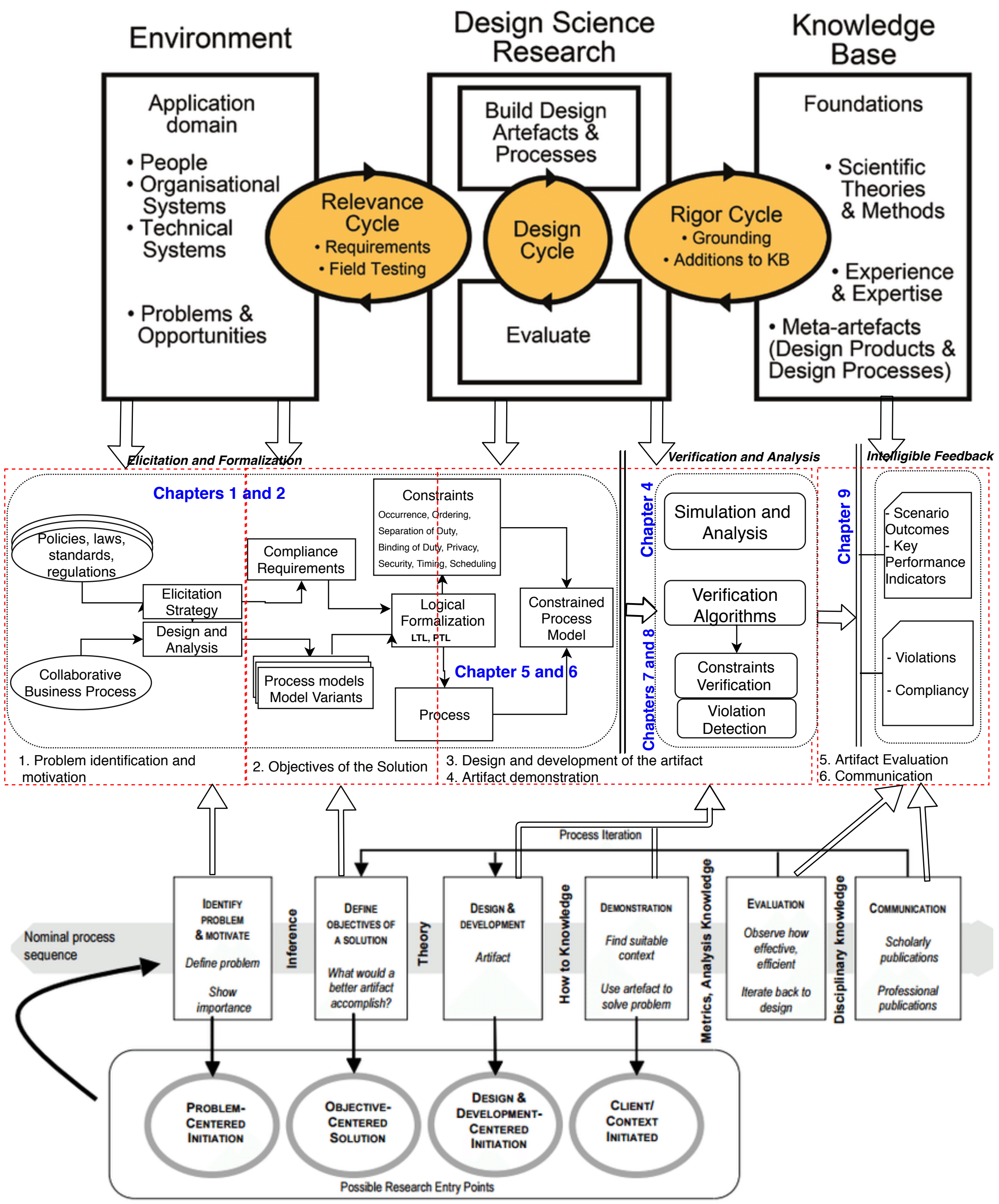

Environment
Application domain
• People
• Organisational Systems
• Technical Systems
• Problems & Opportunities
Design Science Research
Build Design Artefacts & Processes
Design Cycle
Evaluate
Relevance Cycle
• Requirements
• Field Testing
Rigor Cycle
• Grounding
• Additions to KB
Knowledge Base
Foundations
• Scientific Theories & Methods
• Experience & Expertise
• Meta-artefacts (Design Products & Design Processes)
Elicitation and Formalization
Verification and Analysis
Intelligible Feedback
Chapters 1 and 2
Policies, laws, standards, regulations
Elicitation Strategy
Design and Analysis
Collaborative Business Process
Compliance Requirements
Process models Model Variants
Constraints
Occurrence, Ordering, Separation of Duty, Binding of Duty, Privacy, Security, Timing, Scheduling
Logical Formalization
LTL, PTL
Constrained Process Model
Chapter 5 and 6
Process
Chapter 4
Chapters 7 and 8
Simulation and Analysis
Verification Algorithms
Constraints Verification
Violation Detection
Chapter 9
- Scenario Outcomes
- Key Performance Indicators
- Violations
- Compliancy
1. Problem identification and motivation
2. Objectives of the Solution
3. Design and development of the artifact
4. Artifact demonstration
5. Artifact Evaluation
6. Communication
Process Iteration
Nominal process sequence
Identify Problem & Motivate
Define problem
Show importance
Inference
Define Objectives of a Solution
What would a better artifact accomplish?
Theory
Design & Development
Artifact
How to Knowledge
Demonstration
Find suitable context
Use artefact to solve problem
Metrics, Analysis Knowledge
Evaluation
Observe how effective, efficient
Iterate back to design
Disciplinary knowledge
Communication
Scholarly publications
Professional publications
Problem-Centered Initiation
Objective-Centered Solution
Design & Development-Centered Initiation
Client/Context Initiated
Possible Research Entry Points

Figure 3.5: Research Process mapped into Design Science Steps [116] and Research Cycles [74, 72]

Chapter 4

Compliance Requirements Definition, Analysis and Simulation

"Control leads to compliance;
autonomy leads to engagement."

Daniel H. Pink

4.1 Introduction

In chapter 2, related work presented the state of art in regulatory compliance verification. This chapter is concerned with requirements definition, variability and assessing the effect of changes in policies and regulations over business processes. The chapter is structured as follows: Section 4.2 introduces the pick and pack use case as business process, and its applicable policy and regulatory requirements to be complied with. The use case forms a basis to illustrate the different concepts in the chapter. For example, the variability of policies in relation to process model variability. The variability of compliance requirements according to constraint categories is presented in section 4.3. In section 4.4, the second use case describing a car insurance trade process is introduced. Section 4.5 presents the simulation analysis technique to assess the impact of changes and variations in policy over the business process and how they can be exploited to benefit process optimisation. The simulation scenario outcomes support informed decision making on the best coarse of action to take.

4.2 The Pick and Pack Use Case

This section describes the details of use case 1 as a business process for demonstrating the concepts. The business process as shown by the model in Figure 4.1 is based on a giant supermarket with a chain of stores across Europe and parts of Asia. To create orders,

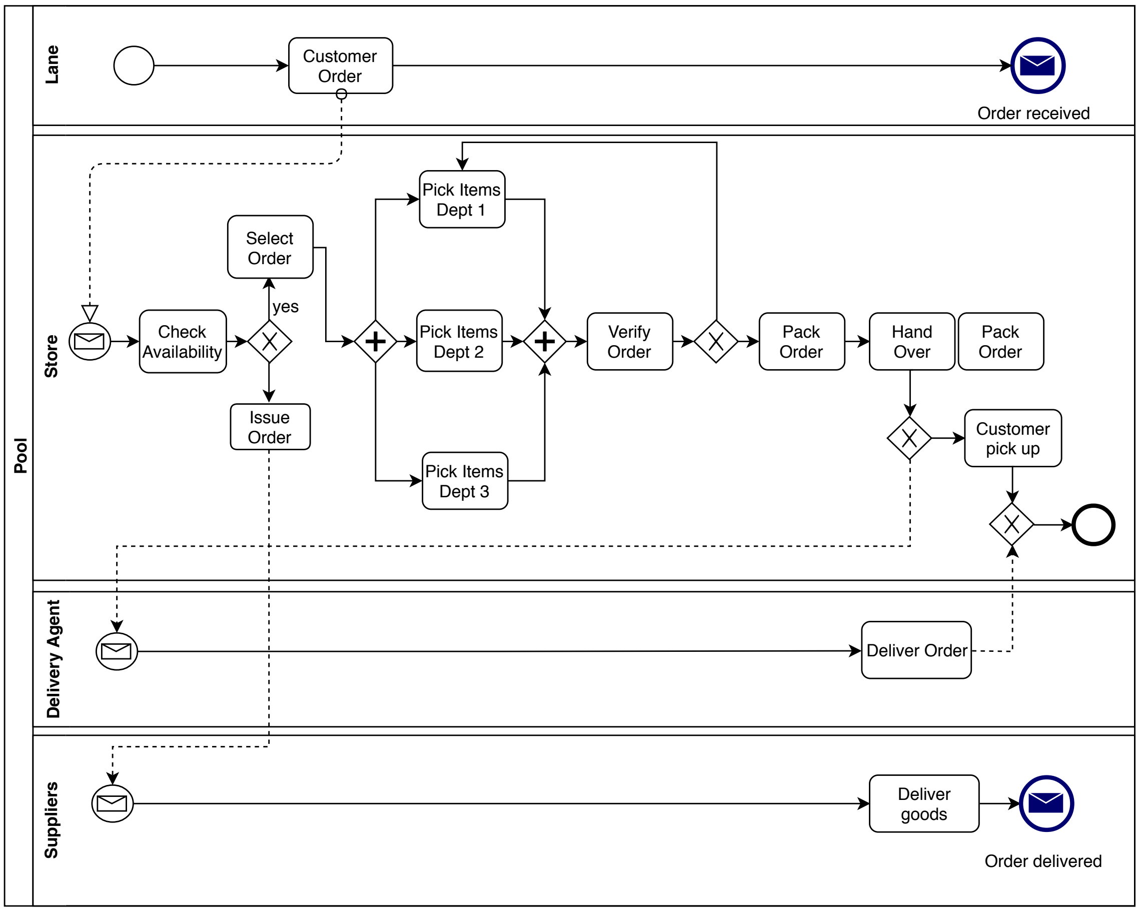


Figure 4.1: Pick and pack business process model

customers register on the store's online system. Once a customer order is received, a notification is received at the store while the customer receives a confirmation. At the Store stock levels are checked for item availability. Where stock is below threshold, a purchase order is issued to the supplier, otherwise order processing progresses. A staff selects an order, picks items and packs the order. Before packing, order is verified for conformity with order details, and after its handed over to customer service. One or more staff may be assigned to an order depending on its size. For items that are out of stock, the order is suspended for a period until stock is available. An item can be substituted with another (e.g. substituting fresh vegetable item with tinned vegetables). Supervisors can contact customers to seek opinion either to wait, change or cancel order in case items cannot be substituted. A customer can cancel an order delayed beyond a specific time. Ready orders are either picked up by the customers or delivered by store. For further understanding, the following assumptions are made;

a. The process model is adopted by stores of different size and capacity.

b. Stores are in different regions where different laws and regulations apply.

c. Stores vary the general model into variant models to suit local policies.

4.2.1 Applicable Policy Requirements

The pick and pack business process is subject to comply with a set of policies and regulations which form constraints that restrict it to specific behaviour and determine how the operations are conducted. The relevant policies are consolidated into a set of policy requirements as exemplified in Table 4.1. They are presented according to the constraint categories discussed in chapter 2. The requirements are generic to permit variations by individual stores.

Table 4.1: General Policy Requirements

Categories	Requirements
Control Flow	Some activities can be combined and executed together depending on store size. E.g. Pick items and pack items. Pack Order immediately follows verify order. Ready orders are either picked by the customer or delivered by the store. Delays are communicated to customers. Pick items is repeated until all items are picked. Notify customer order details immediately after submission.

Continued on next page

Categories	Constraints
Data flow	Customers register on the system before creating orders. Users must be authenticated to access system. System must be up to date with all relevant data. Customers can track their orders via the System. Access to customer data is restricted by privacy constraint. Bulky orders e.g. with orders above £5000 can pay by cheque.
Resource	Resources are assigned to tasks. Resources must be uniquely identified and authenticated. Where resources are assigned work based on shifts, access to data is also based on shifts. Resources like packers and pickers are binding of duty constrained. Some resources like pickers are restricted from executing some tasks e.g. verify order. Some resources like Verifiers are Separation of duty constrained. Some tasks like Hand over order can be delegated.
Time based	System must be available 24/7. Each task is time bound and the total process duration is aggregated from task durations. Some tasks cannot be delayed for more than one hour. Resources are allocated according to time shifts e.g. day shift or night shift.

4.2.2 Applicable Regulations

Besides internal policies, the following relevant external regulations apply;

- The Sarbanes Oxley Act and Basel III with requirements for separation of duty and biding of duty.
- The GDPR with requirements for security and and data privacy.
- The consumer protection Act 2015 UK specifies service level requirements to protect consumers.
- NIST - National Institute of Standards and Technology

The internal policies are established as operational guidelines. They can be varied by stores to suit specific requirements as long as they do not violate the reference policy or regulations. To make the variations, a store considers its size in terms of number of employees, average order quantities and size of stock. Some stores are extra-large characterised by high order volumes, segmented departments and designated employees for each department. Others are small convenient stores with fewer employees and fewer volumes of orders. The variations cater for a store's specific policy requirements suiting operational capacity, business objectives, national laws and standards.

Policy variations lead to process variants configured from the general process model into a process that meets specific operational requirements of a store. Overtime, a process family results [12, 64, 61]. The general process must be flexible and adaptive to permit configurations and variations. For example, a variant process model for store (A) is realised by configuring and individualising the general model to a specific model expressing specific requirements for that store (A). To ensure that both the specific and general requirements are exhibited in the behaviour of the process variant, it is necessary to verify for compliance between both models. This is achieved using different verification techniques such as simulation and model checking.

4.3 Variability of Policy Requirements and Process Variability

Following earlier categorisation of policies and regulations (Chapter 2), this sub section describes how to achieve different versions of the extracted policy requirements (Table 4.1). The illustrations are based on Use case 1 to provide factors for variations;

4.3.1 Variations based on Control Flow Requirements

Control flow perspective specifies the ordering relations among units of work that compose a business process which may be atomic or composite process activities or tasks . A composite activity involves sub activities whereas an atomic one is a single action activity. Variations in control flow policy requirements for the pick and pack process are based the size of the store; Small stores where some activities deemed unnecessary are skipped. Alternatively new ones may be added depending on the requirements and context. Below are sample variations.

1. Policy requirement - item substitution: In case an item is not in stock, the following requirements apply.

    (a) Contact customer for consent to substitute item.

    (b) Response must be received in 2 hours.

    (c) Price of substitute item should match original item price.

    (d) Substitution should not attract additional costs.

    Variation: Above policies can be varied accordingly. For instance, a store may not contact customer to substitute item but avail substitutes during order delivery or order pick up. The customer can accept or reject substitute.

2. Policy requirement - item return: Items are returned to the store by the customer only if; item is a defect e.g. broken or rotten, item does not meet the quality specified or differs from what is described, less in quantity etc. Policies below apply on returne item replacement or compensation;

(a) Items should not have been tampered with e.g. electronic items.

(b) Item is returned in original branded pack e.g. Box.

(c) Required duration to return item differs for each category of the item e.g. fresh items are returned in 24, clothing and shoes have a 30 days return period. Other items are guaranteed for longer periods like 6 months up to a year.

(d) The item is returned with original sales receipt or invoice.

(e) Items must be returned to the store from which they were picked.

Variation: The policy on return of items can be varied in different ways. Some stores may have extended duration based on exceptional terms and item type. For instance, bulky customers who are also re-sellers can return items past ordinary return duration.

4.3.2 Policy Variations based on Data flow Requirements

Policy and regulatory requirements related to data and data flow constrain the information entities consumed or produced during process execution. Data is collected and managed in two forms i.e. data related to the process that supports the control flow of the process, and data required for maintenance of the workflow system. Policies considered here concern variable data that is accessed or produced during process execution. Following the pick and pack process the following data policy requirements apply;

(a) Only registered customers can make orders.

(b) Users must be authenticated by the system.

(c) Order payment must be committed before order confirmation

(d) Privacy is observed for all customer data as required by GDPR

(e) Access to customer data must be authorised.

(f) Basic data is accessible and available for all users.

(g) Order catalogue is accessible to authorised users

(h) Ordering system should be up to date with data about stock.

Variation: Variations should not violate or compromise security and privacy. For instance;

(a) Stores can allow guest accounts for unregistered customers to make orders. For example, unregistered customer can browse and pay for items as long as they can provide addresses for delivery or can identify themselves while picking the order.

(b) Payment processing method may be varied in relation to type of customer e.g. bulky customers can pay on delivery while ordinary customer make payments made upfront.

(c) Customers can only be contacted when item is unavailable or during delivery.

(d) Access control and authorisation variations are based on what data to access and period of access. E.g. privacy on customer data

4.3.3 Policy Variations based on Resource flow Requirements

The resource perspective relates to the actors that execute process activities. They are expressed in form of roles of humans and applications. Policy requirements in this case describe the assignment requirements of resources to tasks. The assignment requirements include;

(a) Binding of duty for select order and pick items activities.

(b) Separation of duty for pick items and verify order activities.

(c) Delegation of tasks between users

(d) Role hierarchy e.g. supervisors has access permission for all supervised activities and users.

Variation: Like for control flow requirements, resource based requirements variations are based on size of store, volume and frequency of orders, sales seasons e.g. peak seasons like Christmas and Easter, job schedules e.g. temporally staff or permanent staff assigned, duration of activities and staff schedules. Due to these factors, variations in the policy requirements can be made in the following ways;

(a) Resources assigned for activities like BoD and SoD may be vary depending on the competency and skills of the role player.

(b) SoD resource assignment may be based on the trust, experience and hierarchy of the role player.

(c) Access control and authorisation may be based on the trust and level of hierarchy a resource holds.

(d) The principle of role hierarchy in view of access control and authorisation may not apply.

4.3.4 Temporal Policy Requirements Variations

Temporal requirements relate to time requirements for executing activities, resources and data access. Temporal requirements do not exist independently but track time constraints for control flow, resource and data requirements and relations between them. e.g. when a task should be executed and for how long. Temporal requirements also constrain resources, e.g. reducing task duration may mean increasing resources or fewer resources may have to be employed for extra hours to complete the task with for a longer duration.The scheduling of duration specifies when tasks can start and end expressible in minutes, hours and days as units. Stores can vary scheduling of both tasks and resources based on local demands. For example, possibility to process an order in a duration of less than six (6) hours, relaxation of item return deadlines, e.g. beyond 24 hours or 30 days in exceptional cases, accepting customer order adjustments beyond the stipulated two hours. Table 4.2 presents the general Pick and pack constraints derived from specified requirements and organised according to constraint categories specified in Chapter 2. It demonstrates a consolidated organised set of requirements from the different sources and relations among them. This simplifies the next step of simulation and analysis.

Table 4.2: General Requirements and Constraints for Pick and Pack Use Case

Category	Requirements	Constraints	Dependency	Related Policies
Control flow	Every order is selected by the actor to start order processing	Initial activity	Select order	Assign resources actors as BoD
	Every order must be verified	Existence of verify order	Pick items	Data and temporal requirements
	Pick items is repeated until all items are picked	Repetition	Select order	Data and temporal requirements
	Notify customer order details immediately after submission	Authorise contact to customer	Receive Orders	Temporal requirements on notifications
	Communicate delays to customers	Authorise contact to customer	Pick items	Temporal requirements
	Order pick up or delivery follows completion of previous activities	Chained precedence	All	Resource, data and temporal requirements
Data flow	Customers register on the system before creating orders	Privacy of customers	Select order	Security and data requirements
	Resource actors authenticated to access system	Authentication	Login	Grant data access accordingly requirements
	System must be up to date with all relevant data	Data availability and accessibility	Identification and authorisation	Authentication and privacy be observed

	Customers can track their orders via the System	Data availability and accessibility	Customer notification	Authentication and access requirements
	Access to customer data is restricted	Authentication and data privacy	Verify order	Assigned resource actors
	Bulky orders e.g. above £5000 can be paid by cheque or other means	Conditional data constraint	Select order	Data flow requirements
	Bulky order customers do not need to pay upfront	Conditional data constraint	Customer login	Data flow requirements
Resource	Resources actors are assigned according to SoD requirements	SoD	Accept tasks	Authentication requirements.
	Resources are uniquely identified and authenticated	Authentication	User identification and authorisation	Data privacy requirements.
	Resource assignment based on shifts data access is assigned accordingly	Authentication and data privacy	Store	Temporal resource assignment requirements
	System must be available 24/7	Availability and accessibility	-	-
	Resources actors are assigned according to BoD requirements	BoD	Select order	Temporal resource assignment requirements
Time based	Maximum order processing duration is 6 hours from submission.	Duration	Select order	Resource and data assignments.
	Fresh items must be returned within 24 hours	Duration	Complaint handling	Resource and data.
	Non-fresh items bear a 30 days return	Duration	Complaint handling	Resource and data.

	Orders delayed beyond reasonable time can be cancelled	Instance duration	Select order	Resource and data assignments.
	Complaints are raised and handled within 7 days	Validity	-	Resource and data assignments.
	Complaints or returns submitted after allowed period are rejected	Validity	Order handover	Resource and data assignments.
	Customer order changes are permitted within 1 hour after order submission	Validity	order submission	Resource and data assignments.
	Store order changes are communicated to customer in less than 2 hours	Validity	Order submission System	Resource and data assignments.

4.4 Use case 2 - The Car Insurance Trading Process

The use case is a car insurance trading process adapted from [166]. The collaborative business process is between 5 key partners i.e. policy holder, Euro Assist Company, Lee consulting services, AGFIL and the garage. The actions of the stakeholders and the process flow is summarised in Table 4.3 while Figure 9.2 is the illustrative process model.

Table 4.3: Actions of collaborating parties in the Insurance Trading Process

Party	Actions
Policy holder	1. The policyholder phones Euro Assist to re- port the car damage. 2. The policy holder sends information to Euro Assist. 3. The policyholder needs to return the claim form to AGFIL during reporting the car dam- age 10 days. 4. The policyholder must send the car to the garage during reporting the car damage claim 1 day.
AGFIL	1. AGFIL need to send the claim form to the policy holder during AGFIL knows the car damage claim 0.6 days. 2. AGFIL need to forward the claim to Lee Consulting Services during the car damage claim receiving 1 day. 3. AGFIL pays the repair cost to the garage during the car damage claim received 30 days.

Europ Assist	1. Europ Assist immediately assigns a garage for the policy-holder. 2. Europ Assist immediately notifies the claim to AGFIL.
Lee Consulting services	1. Lee Consulting Services contacts the garage during the car damage claim received 1.5 days. 2. Lee Consulting Services assigns an assessor to inspect the car if the repair cost more than USD 500 during the car damage received 1.7 day. 3. Lee Consulting Services agrees the garage to repair the car during the car damage claim received 3.5 days. 4. Lee Consulting Services forwards invoices to AGFIL during the car damage claim received 6 days.
Assessor	1. The assessor inspects the car for Lee Consulting Services during the car damage claim received 3 days. 2. The assessor sends a new repair cost to Lee Consulting Services during the car damage claim.

Similarly, using the case details it is possible to make variations in policy requirements for the insurance trading process to suit specific application. The case is revisited in Chapter 9 and used to support evaluation of the designed verification algorithms.

4.5 Changes in Policy and Regulations

Amendments in policies and regulations have direct effect on the business and its existing business processes. For example, an amendment in tax policy may specify new ways through which organisations report tax compliance, or a revised requirement specifying how companies should report their financial status etc. Some changes may be internally sourced as a way to improve workflow, reach a new market or satisfy a particular market demand. Such changes may cause an organisation to modify its entire process or part of it to achieve compliance. Achieving regulatory compliance by re-engineering business processes is a non-trivial task. In many cases, organisations have to hire new employees like compliance officers, this raises a financial burden. For example, the revised GDPR data privacy requirement emphasises roles new roles that should exist in organisations. Regardless of the magnitude of the change or the size of the business process, it is imperative to follow a formal method to identify changes in the policy, the components of the process that are affected by the amendments, commit the amendments and check the process to ensure that compliance is attained.

Compliance documents are written in natural languages with associated ambiguity. The ambiguity leads to false interpretations, misunderstandings and confusion. Moreover, regulations are stated in a prescriptive manner, i.e. they specify what is required but are silent on how it should be achieved. As a consequence, organisations have perceived compliance as a tedious burden and a complex task especially where skills or automated compliance tools are not available to support enforcement and verification. This thesis presents a compliance approach to support;

- Elicitation of new compliance requirements arising out of the changes in policy and regulations

- Formalisation of the requirements into compliance constraints and,

- Checking and verifying compliance of the business process with the constraints.

An initial step towards compliance management is the ability to identify the relevant policies and regulations that the business process should conform with. These are derived from the relevant regulations as the previous sections have shown. The identified changes are formalised and used check the business process for conformity. Changes committed in the process can be a source of non-compliance. Checking the impact of policy and regulatory changes over the business process is essential to determine its compliancy. Verification techniques like theory proving, model checking and simulation. These techniques were introduced in chapter one section 1.4. In the following section, simulation technique is used to assess the impact of the changes in policy and regulatory requirements over the performance of the business process.

4.6 Simulation Based Analysis of Changes in Policy and Regulations

4.6.1 Simulation

Simulation refers to activities undertaken to imitate operations of a system using a model. The model is then studied through configuration and experimentation to understand the actual properties and behaviour of the system or its components. Besides the general advantages of simulation like elimination of diverse bottlenecks, flexibility and resource optimisation, simulation-based analysis for policies and regulations is useful to assess the different options of implementing new or modified policies to a business process by providing potential impact over its elements, inputs and outcomes to support. This way, informed decision making is supported in a less costly means to improve and optimise a business process. For instance, resource usage and allocation are optimised by simulating different scenarios of allocation of staff by varying their numbers or other parameters, and then assess the impact over service times in the process. Through mimicked behaviour of a process's components and their interaction, it is possible to predict and understand performance of the whole system, assess different alternatives to provide resource capacity or innovative ways to improve performance of the process.

Simulation involves a set of steps; identify the problem, formulate the problem, collect and process real system data, develop the model, validate the model, document the model for future use, perform simulation runs and, analyse and present results [101, 33, 32, 150, 151]. Besides traditional simulation methods based on mathematical models, several modern tools and software are used to simulate processes in different industry sectors. Key examples in the business process management industry are tools like Bizagi simulation studio, Simul8, Protos, Ingrid Cloud and AnyLogic among others. In manufacturing, tools like SIMPROCESS and FlexSim are used to simulate manufacturing and engineering processes. These tools present similar characteristics like graphical interfaces, graphical plots, animations and dashboards which makes them easy to use.

For illustration purposes, Bizagi simulation modeller is used as a simulation tool in this thesis. Bizagi was preferred over other tools because it offers ease of use, flexibility in model design and it has wider online user community that provides rich knowledge for support. To proceed with the illustration, a modified use case of the pick and pack business process is reintroduced; Figure 4.2 is the abstracted model from the original one in section 4.2 to fit simulation illustrations due to space limitation.

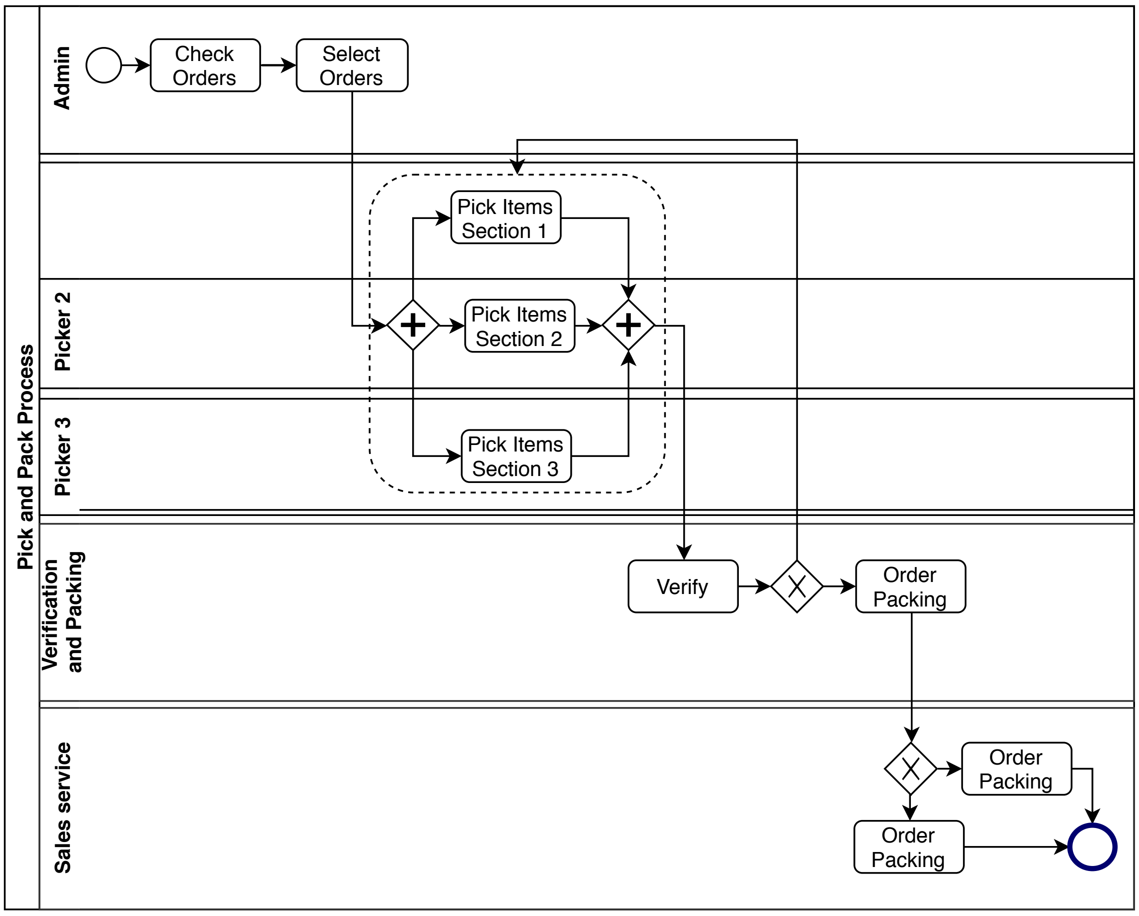


Figure 4.2: Abstracted Model of Pick and Pack Process

4.6.2 Simulation Scenarios - Scenario 1

Scenario 1 Baseline Information

The simulation scenario begins from the point when orders arrive at the store. The following assumptions are considered as policies to guide operations:

- Stock is automatically replenished i.e. the store never runs out of stock.
- The store has three sections from which items are picked, i.e. Section 1, 2 and 3. Staff are assigned to a single section.
- Staff who pick items do not verify order.
- Orders are processed as they arrive.
- Staff work for one shift a day with a single day off in a week.
- On average, 10% of the orders do not pass verification.
- Pick items in section 1 is allocated more processing time (50% of 30 minutes) due to high item orders from that section.
- A big percentage (80%) of the orders are picked by customers, only 20% is delivered.
- The cost for the delivery van is fixed at £2 for each delivery. Other resource costs are charged hourly.
- It takes two hours to process an order from select order to Hand over.

Based on the above assumptions, data in Table 4.4 is used for simulation of scenario 1. In this scenario, 1000 instances are simulated representing orders processed for a period of 30 days. The intended objective is to analyse the business process using base line data and project possible outcomes based on operational policies. The outcomes form a basis upon which policy variations are bench marked and compared. Additionally, the analysis enables identification of key performance indicators in resource utilisation.

Table 4.4: Scenario 1 Base line Data

Activity	**Waiting time (Min)**	**Processing time (Min)**	**Resource**	**Assigned quantity**	**Default Quantity**	**Hourly Unit cost**	**Fixed costs**	**Shift availability**
Check order	0	10	Supervisor	2	2	15	-	1
Select Order	1	5		3			-	
Pick Items Sec 1	3	16		1			-	
Pick Items Sec 2	3	7	Pickers	1	6	8.5	-	5
Pick Items Sec 3	3	7		1			-	
Pack order	4	15	Packers	2	4	9	-	3
Verify order	3	10		2	-			
Hand over	1	5	Assist. Supervisor	2	2	12	-	2
			Delivery staff	1	2	10	-	1
Delivery	10	40	Delivery van	1	1		2	1

(a) Scenario 1 Outcome Analysis

Table 4.4 shows the detailed data required in simulation scenario 1 where the activities are listed with their waiting time, processing time, assigned resources, their quantity and costs. Based on this data, the simulation runs of maximum arrival count =1000 (process instances) yield the model in Figure 4.3 and generate further data to support the analysis.

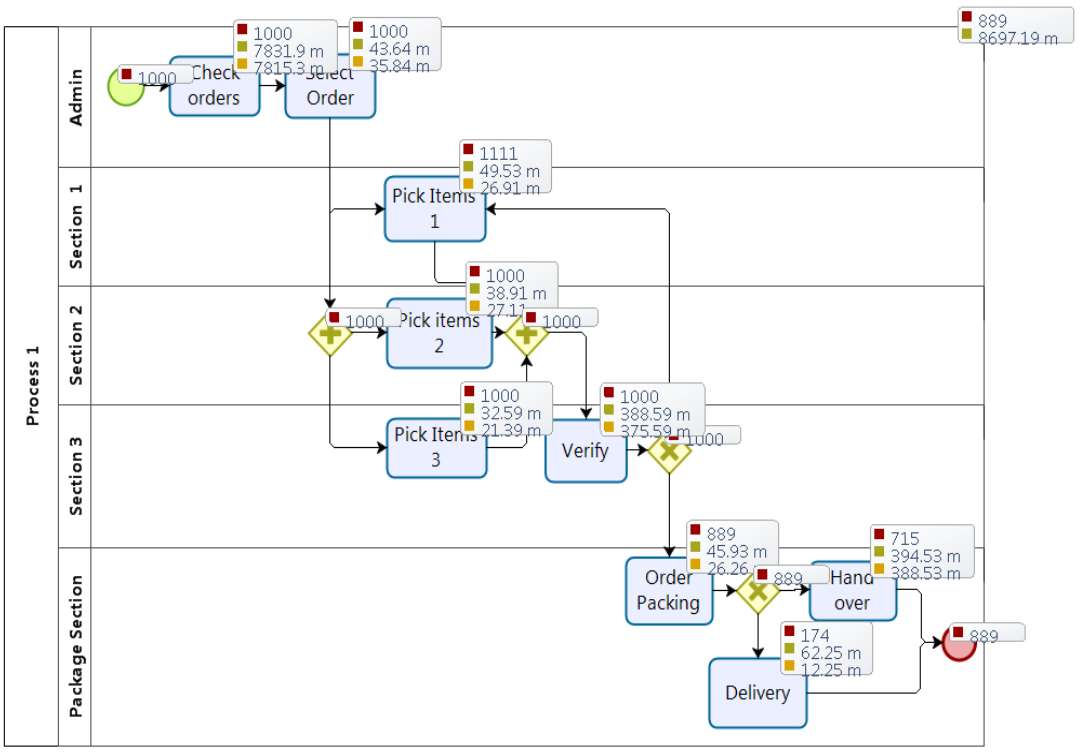


Figure 4.3: Model Output Summary

Presented in Figure 4.3 is a summarised simulation analysis model showing executed instances for each activity and the total time spent to execute all activity instances for all the 1000 cases simulated. To note from this figure is the variance between the instances at both start event (1000) and end event (889). The difference of 111 cases is accounted for by the cases that are not successfully verified. These are looped back to Pick items 1 task (note the 111 instances at this task and in Table 4.6). Further analysis is described in the next section based on varied inputs.

(b) **Scenario 1 - Resource Utilisation Analysis**

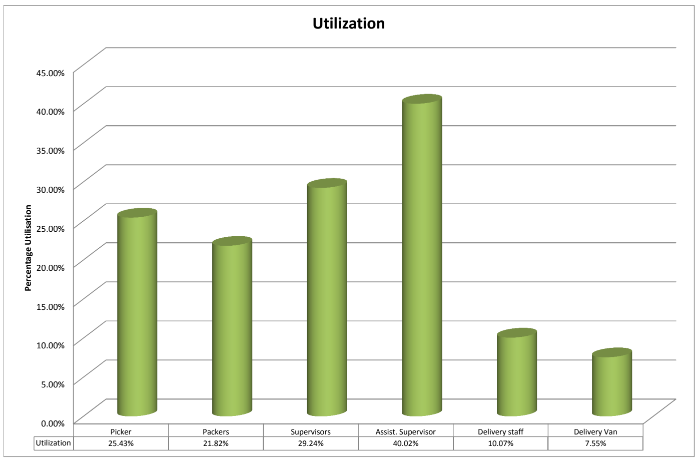
Resources are assigned to tasks following policies and assumptions stated previously. Output is presented in Table 4.5 showing the Assist. Supervisor as the most utilised resource (at 40.02%) with a cost of £6,916 per month. However, the utilisation of resources is below average at about 40% highest and 7.55% for the lowest (delivery van). In addition, the scenario shows that Pickers are the most cost intensive resource requiring £8688. Table 4.5 presents a summary of the of the complete outcome from the simulation of resource utilisation.

Table 4.5: Resource utilisation data for scenario 1

Resource	**Utilisation**	**Total fixed cost**	**Total unit cost**	**Total cost**
Picker	25.43%	-	8688.5	8688.5
Packers	21.82%	-	5067.3	5067.3
Supervisors	29.24%	-	5000	5000
Assist. Supervisor	40.02%	-	6916	6916
Delivery staff	10.07%	-	1450	1450
Delivery Van	7.55%	348	0	348

A further summary of the resource allocation and usage is presented in Figure 4.4 in which the Assistant supervisor is depicted as the most utilised resource followed by the supervisors. Therefore, policies regarding allocation and utilisation of these resources must be taken with regard to their capacity or cost requirements.

With regard to temporal requirements, Table 4.6 presents time based simulation outcomes. The data shows that it requires 94 minutes to process a single instance of an order while 8,724,415.9 minutes to process 1000 orders given the same resource assignments. The durations fit well with the projections in the assumptions, e.g. 94 minutes are less than the projected two (2) hours. The column for Min. time shows the least time



	Picker	Packers	Supervisors	Assist. Supervisor	Delivery staff	Delivery Van
Utilization	25.43%	21.82%	29.24%	40.02%	10.07%	7.55%

Figure 4.4: Summary of Resource Utilisation for Scenario 1

spent executing an activity by a resource. For instance, the minimum time for Pick Items from section 1 is 21 minutes, the maximum is 653.4 minutes (approximately 11 hours) while the average is 49.5 minutes. The maximum time is the worst case scenario. The total time required to execute all cases is 55030.1 minutes. This would be too much time spent on order processing yet resources are not fully utilised. Therefore, we proceed to make adjustments in policies for assigning resources and then assess their impact on the business process. This leads to the next simulation scenario where the inputs are adjusted.

4.6.3 Simulation Scenarios - Adjusted Scenario with Policy variations

This scenario is intended to present an analysis of the impact of adjustments in policy over the business processes. Following from the baseline data in scenario 1, policy variations regarding resource allocation, quantities and costs form input into the simulation model. The output is analysed and compared with base line data. The variations in policy are marked in Bold text:

- The store has three sections, **staff can cross between sections**.
- Orders are processed on first come first serve basis.

Table 4.6: Task based output in relation to time

Name	**Instances completed**	**Min. time (m)**	**Max. time (m)**	**Avg. time (m)**	**Total time (m)**
Process 1	889	94	16607	8697.1	8724415.9
Check orders	1000	10	15601	7832	7831900
Select Order	1000	6	672	44	43635.9
Pick Items 1	1111	21	653.4	49.5	55030.1
Pick Items 3	1000	10	642	32.5	32588.2
Verify	1000	13	740	388.5	388587.8
Hand over	715	13	743	394.5	282091
Pick items 2	1000	10	648	38.9	38914.4
Order Packing	889	19	619	45.9	40836.1
Delivery	174	50	178	62.2	10832.1

- **The store operates in two work shifts a day. A shift is 8 hours.**
- **With the introduction of second shift more staff were hired**
- A reduction in verify order errors to 5%.
- Picking items from section 1 is allocated more processing time (50% of 30 minutes) perhaps because of the nature of goods.
- **A 70% reduction in orders picked by customers while 30% is delivered.**
- **The delivery van cost is fixed at £1. Other resources are charged per hour.**
- **A projected increase in staff minimum wage from £ 8.5 to £ 9.5 per hour.**

Based on the assumptions above and adjusted inputs, data in Table 4.7 is used to generate a variant simulation model to support further analysis. The aim is to support process optimisation through what if analysis of different scenarios based on policy variations, assess the outcome and its impact on the business process. The adjusted inputs result into a simulation model in Figure 4.5 with cast data from 1,000 instances running for 30 minutes.

Table 4.7: Adjusted Scenario Data

Activity	**Waiting time (Min)**	**Processing time (Min)**	**Resource**	**Assigned quantity**	**Default Quantity**	**Hourly Unit cost**	**Fixed costs**	**Day Shift**	**Evening Shift**
Check order	0	10	Supervisor	2	2	15	-	2	2
Select Order	1	5		2	8	9	-		
Pick Items 1	3	16		1			-		
Pick Items 2	3	7	Pickers	1			-	3	5
Pick Items 3	3	7		1			-		
Pack order	4	15	Packers	2	5	9.5	-	2	3
Verify order	3	10		2	4	13	-	2	2
Hand over	1	5	Assist. Supervisor	2			-		
Delivery	10	40	Delivery staff	1	2	10	-	1	1
			Delivery van	1	1		1	1	1

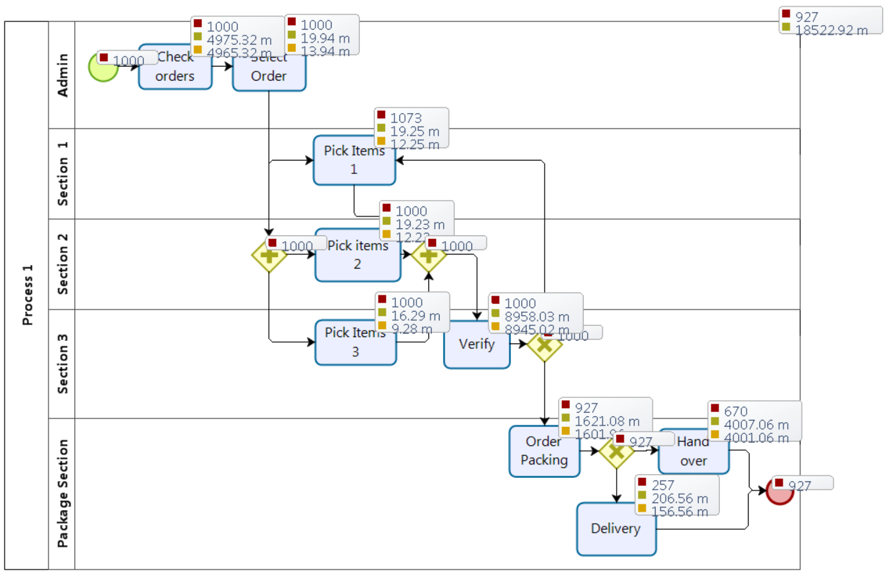


Figure 4.5: Process Model for Adjusted Scenario

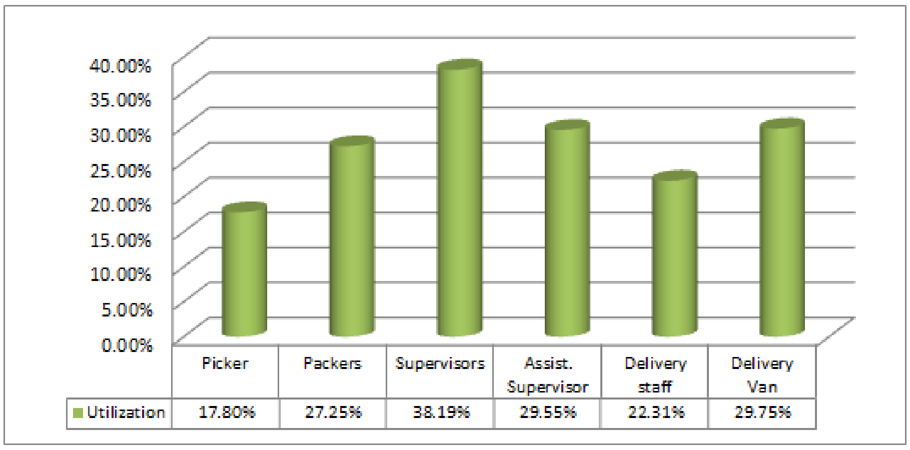
Based on the adjusted inputs, simulation outcomes regarding resource utilisation are presented in Table 4.8 and graphically in Figure 4.6. Both the table and the figure show that the increase in resource quantities has reduced the utilisation percentages and costs for some resources where for others there is an increase. E.g. in Table 4.8 the utilisation percentage has reduced to 17.8% compared to 25.43% in Table 4.5, the same applies to their costs reducing from £8688.5 to £6495.2. Further utilisation reductions are realised for the Assist. Supervisor (from 40.02% to 29.60%) and related costs. The rest of the resources have increased utilisation percentages and the related costs. E.g. the increase in the utilisation of Supervisors from 29.24% to 38.20%. Details of these variances are presented in Tables 4.5 and 4.8 while section 4.6.4 presents detailed comparison between the two scenarios.

Figure 4.6 graphically summarises resource utilisation and indicates supervisors as the most utilised (38.19%) report whereas the pickers are shown as the least utilised (17.80%).

Simulation of time requirements for the adjusted scenario yields data presented in Table 4.9. With the increase in successful instance executions from 889 (Table 4.6) to 927 (Table 4.9), the average execution time increases to 18522.9 minutes from 8697.1 minutes in scenario 1. Besides, the average required execution durations variably increase or de-

Table 4.8: Adjusted Scenario 1 Resource Utilisation

Resource	**Utilisation**	**Total fixed cost**	**Total unit cost**	**Total cost**
Picker	17.80%	0	6495.2	6495.2
Packers	27.30%	0	6214	6214
Supervisors	38.20%	0	8250	8250
Assist. Supervisor	29.60%	0	7375.3	7375.3
Delivery staff	22.30%	0	2141.7	2141.7
Delivery Van	29.80%	257	0	257



	Picker	Packers	Supervisors	Assist. Supervisor	Delivery staff	Delivery Van
🟩 Utilization	17.80%	27.25%	38.19%	29.55%	22.31%	29.75%

Figure 4.6: Adjusted Scenario 1 Resource Utilisation Chart

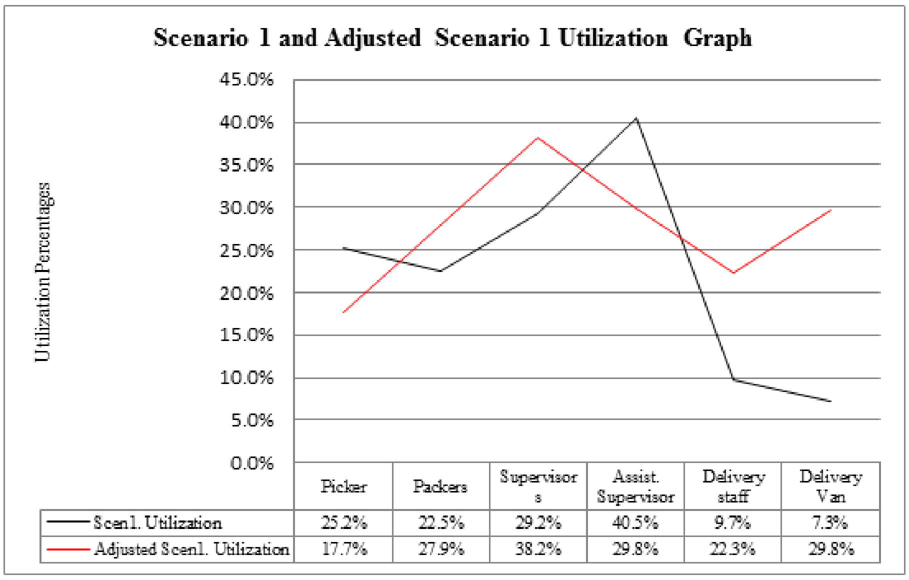
crease as presented in table 4.9 increase in processed instances or increase in assigned resources. The general implication from the adjusted data following from the variations in policies is that the increase/ decrease in resources or their allocation directly affects the number of activities or tasks that would be completed as well as the time that will be spent executing them. This must be considered when policies are to be changed.

Table 4.9: Adjusted Scenario 1 Process Time Sheet

Name	Instances completed	Min. time (m)	Max. time (m)	Avg. time (m)	Avg. time waiting for resource (m)
Process 1	927	338	23311.1	18522.9	-
Check orders	1000	10	9976	4975.3	4965.3
Select Order	1000	6	71	19.9	13.9
Pick Items 1	1073	7	71	19.2	12.2
Pick Items 3	1000	7	65	16.3	9.3
Verify order	1000	283	9983	8958	8945
Hand over	670	6	8753.9	4007.1	4001.1
Pick items 2	1000	7	67	19.2	12.2
Order Packing	927	19	4103	1621.1	1601.96
Delivery	257	50	593	206.6	156.56

4.6.4 Comparison Between Scenario 1 and Scenario 1 with Adjusted Data

To understand the impact of policy variations on the business process, resource utilisation graph and resource cost graphs are used to make comparisons between baseline data and adjusted data. The graphs in Figures 4.7 and 4.8 are used for comparison purposes. The utilisation of resources between the two scenarios in Figure 4.7 does not show a significant difference except for a slight increase in the utilisation of supervisor resource. Figure 4.8 shows the contrary with a sharp increase in the resource cost for the adjusted scenario. Therefore the adjustments have not indicated any positive outcome since the policies have led to increase in resource costs and more order processing time without significant increase in number of orders processed. Because the outcomes are not realistic yet, we proceed with another scenario further adjusting the policies relating to resource allocation and while doubling the number of orders over the same period of time (30 days)

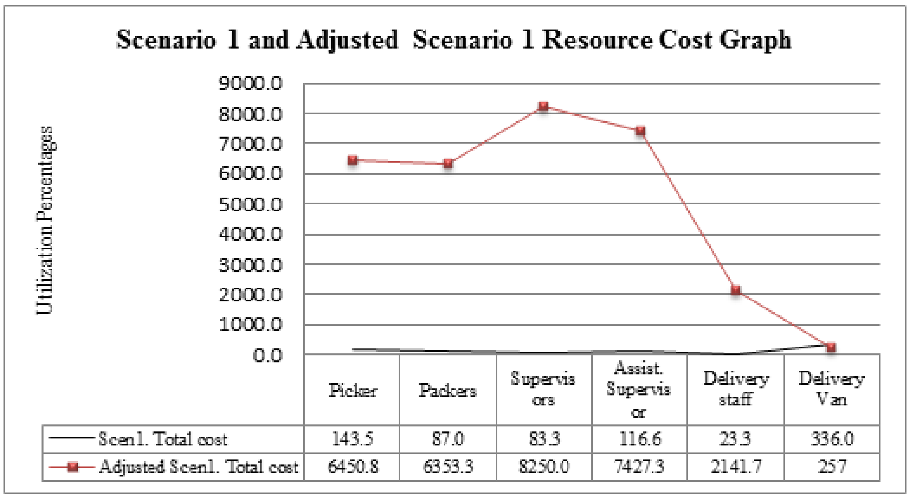


	Picker	Packers	Supervisors	Assist. Supervisor	Delivery staff	Delivery Van
Scen1. Utilization	25.2%	22.5%	29.2%	40.5%	9.7%	7.3%
Adjusted Scen1. Utilization	17.7%	27.9%	38.2%	29.8%	22.3%	29.8%

Figure 4.7: Comparison Resource Utilisation Graph

4.6.5 Simulation Scenarios - Scenario 2

In this section, a new simulation experiment is drawn with new data based on different policies. The purpose of scenario 2 simulation is to analyse and optimise the same busi-



	Picker	Packers	Supervisors	Assist. Supervisor	Delivery staff	Delivery Van
Scen1. Total cost	143.5	87.0	83.3	116.6	23.3	336.0
Adjusted Scen1. Total cost	6450.8	6353.3	8250.0	7427.3	2141.7	257

Figure 4.8: Comparison Resource cost Graph

ness process using different assumptions from scenario 1. In this case, the 2000 instances are used to simulate and project process performance for 30 days. Other assumptions in the scenario 2 follow;

- All sections have equal order requirements i.e. same item quantities are processed for all the departments of the store.
- Same staffing levels are maintained as in scenario 1. However, they are allocated as follows;
  - staff cross between sections to pick items.
  - Staff work all days of the week and an increase of £2 for each staff category is administered.
- Customers are encouraged to use store delivery at a reduced charge of £1. A new delivery van is acquired for the purpose. The percentage distribution between store pickup and store delivery is set at 60% and 40% respectively.
- Errors during verification are projected to reduce to 3% since the supervisor role will be participating (one supervisor).
- Two (2) packing staff can help during the peak hours of hand over.

Based on the above assumptions, the data in Table 4.10 is used to support the simulation and analysis of outcomes of scenario 2.

Table 4.10: Scenario 2 Data table

Activity	**Waiting time (M)**	**Processing time (M)**	**Resource**	**Assigned quantity**	**Default Quantity**	**Hourly Unit Cost**	**Fixed Costs**	**Shift Availability**
Check order	0	10	Supervisor	2	2	17	-	2
Select Order	1	5		3	6	10.5	-	6
Pick Items Sec 1	3	16		1	6	10.5	-	6
Pick Items Sec 2	3	7	Pickers	1	6	10.5	-	6
Pick Items Sec 3	3	7		1	6	10.5	-	6
Pack order	4	15	Packers	2	4	11	-	4
Verify order	3	10	Assist. Supervisor	2	2	14	-	2
Verify order	3		Supervisor	1	2	14	-	2
Hand over	1	5	Assist. Supervisor	2	2	14	-	2
Hand over		2	Delivery staff	2	2	12		2
Delivery	10	40	Delivery van	2	2	-	1	2

Scenario 2 Model analysis

Data in Table 4.10 shows the order of activities, their execution duration as well as assigned resources. The simulation is run based on this data for a period of 30 days. The instances are increased to 2000 cases of order processing (instances). The simulation model in Figure 4.9 results and provides a basis for scenario 2 analysis. From Figure 4.9 it can be noticed that using the stated assumptions, 2000 orders cannot be served within 30 days period.

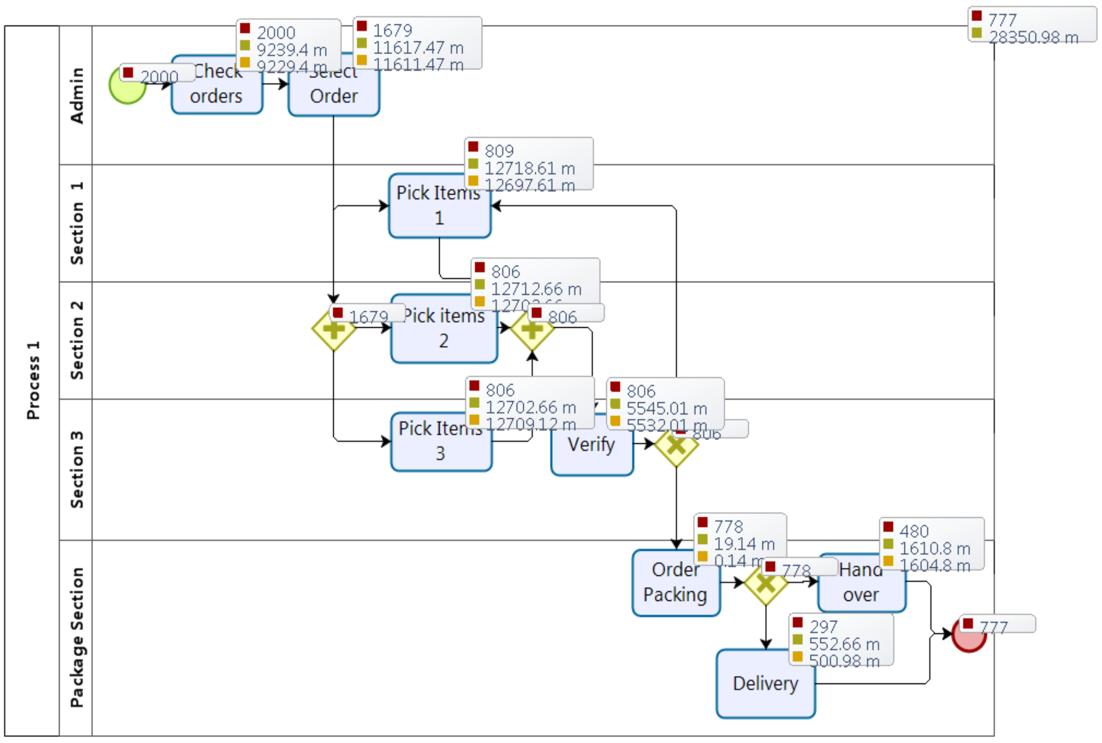


Figure 4.9: Scenario 2 model with incomplete instances

Figure 4.9 shows that by the end of projected period (30 days); only 777 orders out of 2000 orders would be processed as seen at the end event. This can be attributed to staffing levels and allocations, task wait times and execution errors. Besides it could as well be to unrealistic projection period. To optimise order servicing to acceptable levels, the simulation period is adjusted to 55 days. To that effect, another simulation model results in Figure 4.10 are realised.

Even with the adjusted duration, Figure 4.10 shows that maximum projected orders to be

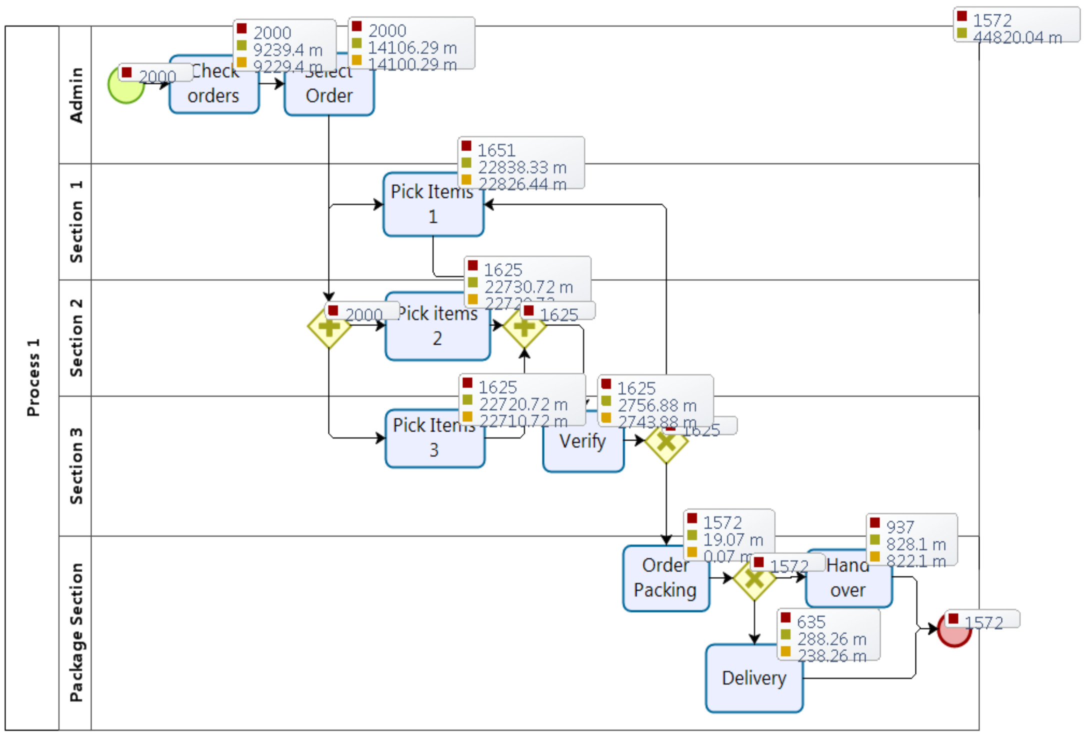


Figure 4.10: Scenario 2 model with adjusted period

serviced cannot be achieved. By the end of 55 days, about 1572 orders would be serviced with a low margin of errors at verify order. Analysis of the resource allocation and usage based on Table 4.11 and resource graph in Figure 4.11 reveals constrained resources. This is a bottleneck limiting the achievement of target. It is advisable to proceed and create another scenario for resource allocation to achieve optimum allocation. In section 4.6.6, we illustrate resource adjustment scenarios. Table 4.11 presents resource utilisation data further summarised by the graph in Figure 4.11

4.6.6 Resource Adjustment Scenario

Analysis of the resource data in Table 4.11 and graph in Figure 4.11 reveals an overly utilised resource of 'pickers' to almost 100%. A decline from 2000 instances of orders at select order task to 1651 instances at pick items is noticed from the simulation model 4.10. This sharp drop is potential pointer to the source of the problem and perhaps explains why all orders cannot be serviced within the projected period. The issue can be

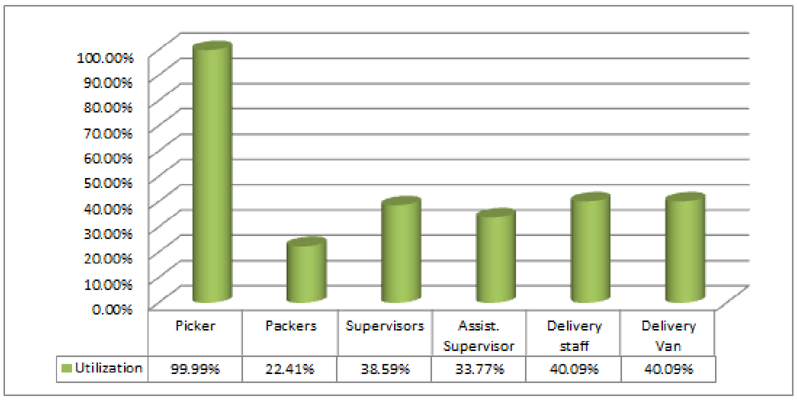


Figure 4.11: Scenario 2 Resource Utilisation Graph

Table 4.11: Scenario 2 Resource utilisation Graph

Resource	**Utilisation**	**Total fixed cost**	**Total unit cost**
Picker	99.99%	0	83150
Packers	22.41%	0	13015
Supervisors	38.59%	0	17319
Assist. Supervisor	33.77%	0	12482
Delivery staff	40.09%	0	12700
Delivery Van	40.09%	635	0

approached with various alternative solutions;

- To reallocate staff from other sections to the picking section especially at pick hours since they are used at less than half capacity as the percentages show.

- Hiring more picking staff.

- Creating 2 or more working shifts since the scenario is based on a day shift calendar of 10 hours. Increasing the simulation duration alone is not a solution since the attempt in previous section did not yield much outcome towards the target. Moreover, we are stretching beyond 30 days which was the initial service target period. This would mean sales are maintained but problems relating to customers are not solved. After several adjustments and simulations runs involving increase in staffing for the pickers to a quantity of 10 and allocating all of them to full time availability and reducing the waiting time for the picking activities, a more feasible outcome is achieved in 26 days servicing 1938 instances. Figure 4.12 shows the model with adjusted resources and model results. However, the process execution time shoots high to 7 hours and 12 minutes. Moreover, utilisation of pickers has been balanced with increased 4 staff though it remains above average as Table 4.12 and Figure 4.13 show. From this analysis, we can note that picking activity is crucial to the execution of the process and achieving its objectives and targets.

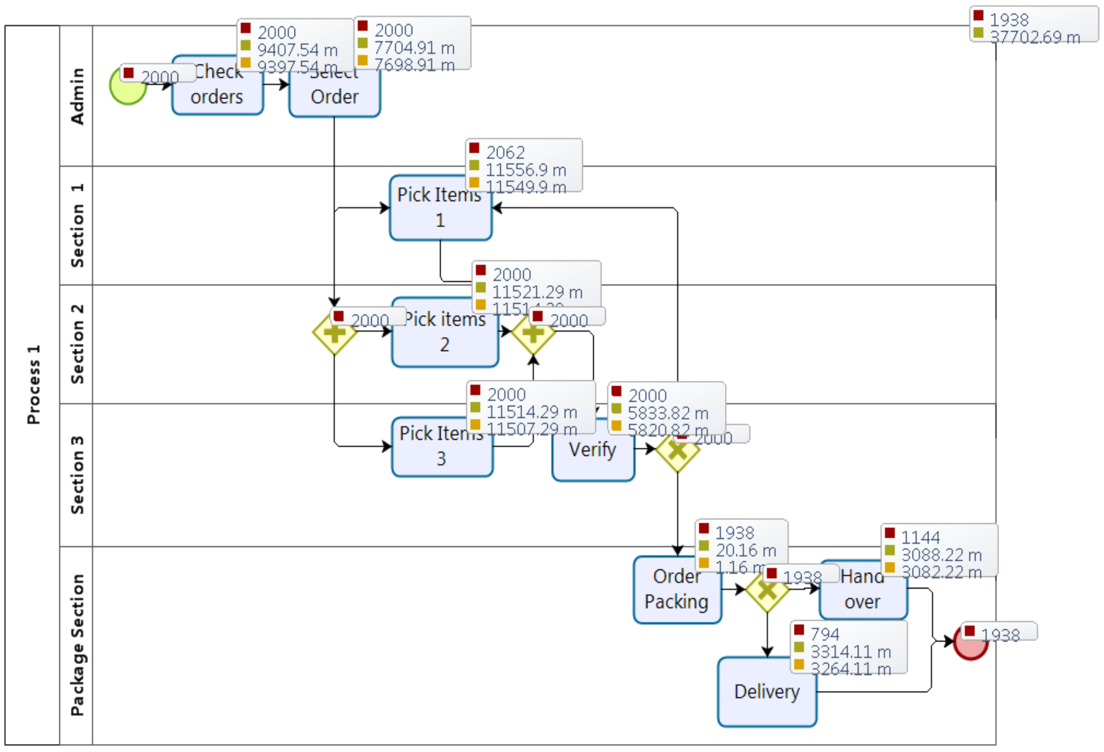


Figure 4.12: Realistic model from scenario 2

4.6.7 Resource Usage Comparison Between Scenarios 1 and 2

Comparison between scenarios 1 and 2 is based on resource actors, their cost and time as Table 4.13 shows. The disparity between the scenarios is traced back to the adjustments made on resource data for Pickers. The increase in resource quantity (staff numbers) facilitated a reduction in task waiting time and processing time for the Pick items task as well as in the entire processing time of the business process. For instance, the increase

Table 4.12: Scenario 2 adjusted output data

Resource	**Utilisation**	**Total fixed cost**	**Total unit cost**
Picker	57.62%	0	79859.5
Packers	27.58%	0	16018.2
Supervisors	41.67%	0	18700
Assist. Supervisor	41.49%	0	15337
Delivery staff	50.13%	0	15880
Delivery Van	50.13%	794	0

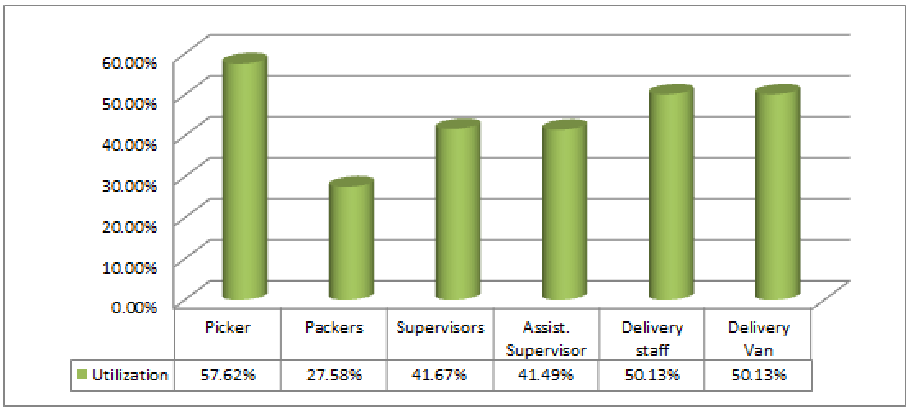


Figure 4.13: Adjusted Resource utilisation Graph for scenario 2

in processed instances from 1752 to 1938. Therefore, informed decision making can be supported using;

- Key performance indicators (KPIs): KPIs refer to measurable values that are used to evaluate and show operational success of a given entity. Based on the simulation experiment above, the key performance indicators for the pick and pack business process resources include the Pickers. This is more especially at peak times when Items are picked, at task waiting times and task processing times.
- Increasing resource quantities at critical tasks/ activities is essential to improve the performance of the process in terms of more instances or orders processed, and overall task execution durations.

Table 4.13: Scenarios 1 and 2 Data

Resource	**Scenario**	**Utilisation**	**Total fixed cost**	**Total unit cost**
Picker	Scenario 1	25%	0	143.5
	Scenario 2	57%	0	79626.8
Packers	Scenario 1	22%	0	87
	Scenario 2	28%	0	16280.4
Supervisors	Scenario 1	29%	0	83.3
	Scenario 2	42%	0	18700
Assist. Supervisor	Scenario 1	40%	0	116.6
	Scenario 2	42%	0	15501.7
Delivery staff	Scenario 1	10%	0	23.3
	Scenario 2	48%	0	15080
Delivery Van	Scenario 1	7%	336	
	Scenario 2	48%	754	

  - Cost correlatively increases with the increase in resources (e.g. staffing costs). Process designers and implementers should be able to base their decisions on cost vs benefit analysis. For example, the cost for pickers overshot from £143.5 for scenario 1 to £79,626.8 in scenario 2 while the fixed costs for adding another van while reducing the delivery charges rose from £336 in scenario 1 to £754 in scenario 2, of course bearing in mind that latter figure covers longer period.

Simulation provides flexibility and predictability based on what if scenarios which supports process thinkers, designers and analysts to easily modify variable process elements and simulate output to support informed decisions on the best courses of action. However, simulation has a set of associated limitations.

- Simulation cannot reveal intricate errors in the business process. In cases where experiments are based on assumed data, the results may be treacherous and unreliable especially when they differ from actual environments.

- Moreover, simulation provides no proof that what is experimented will exactly happen the same way in reality.

Based on the limitations, robust analysis and verification is recommended to reveal further errors in business process models. In the next chapter (Chapter 5), another verification technique i.e. model checking is used to verify process model errors that scale beyond simulation analysis.

4.7 Contractual Requirements Generation

In collaborative environments, several partners combine resources to design and execute collaborative business processes. Besides the partner specific policies and external regulations, collaborative business processes are as well governed by contractual obligations i.e. rules established to guide the collaboration clearly indication partner responsibilities. Therefore, a collaborative business process choreography will have three views of policies and regulations i.e. 1) the local view which describes policies governing internal business logic of private processes, public view describing policies at involving contractual obligations, and 3) the global view composed of regulations guiding process behaviour industry wise [99, 48]. The compliance challenge at this level concerns the propagation of changes from one level across the different levels to ensure consistency, validity and integrity of the process choreography. Some of these challenges are being addressed by [99, 48] through proposed change propagation algorithms.

4.8 Chapter Summary

The chapter demonstrated the impact of changes in policies and regulations over business processes. Changes in policies and regulations are inevitable and happen over time requiring adaptation or modification of existing business processes. In this chapter we have shown how to check and analyse impact of changes in policies and regulations over the performance of a business process. We have used policy variations of the pick and pack use case to arrive at different simulation scenarios. The specific chapter outcomes are as follows; - It introduced the use cases and their requirements as proposition elements for compliance constraints formulation. - Based on the use cases, the chapter showed how the policies and regulations can be varied over data, time and resource constraints using several scenarios managers. - Using simulation analysis technique, the effect of the changes in policy and regulatory requirements over the business processes have been analysed, outcomes assessed against key performance indicators and presented using summarised graphs showing comparisons. The chapter also points out on the different levels of policy and regulatory views typical of collaborative business process choreography.

Chapter 5

Expression and Specification of Compliance Constraints

5.1 Introduction

This chapter presents concepts related to compliance requirements, their expression and translation into constraints. Based on requirements from use case one (Pick and Pack) and citing of examples, an illustration of elicitation and categorisation of requirements is presented in the chapter. To achieve expression and representation of compliance requirements, description logic is introduced and used, whereas the translation of requirements into constraints is enforced by integration of description logic and linear temporal logic to achieve formalised constraints to which reasoning can be applied to achieve compliance verification. The Chapter is structured as follows: Section 5.2 presents the constraints described with illustrative examples, while section 5.3 describes the constraints specific for collaborative business processes. In section 5.4, the expression of unary constraints according to their categories is presented using description logic while in section 5.5 composite expressions are presented. The chapter is concluded with section 5.6.

5.2 Constraints

Constraints restrict processes to specific behaviour as required by policies and regulations. Without the restrictions models would execute any desired behaviour or end users would be at liberty to undertake any desired operations. However, the existence of constraints creates restrictions on process behaviour. A *constraint* is a rule prescribing behaviour as conditions that a business process must conform with. Mandatory constraints must be satisfied by the execution of the model otherwise a violation occurs. The optional constraints may be satisfied or not, their lack of satisfaction does amount to a compliance violation. The choice to execute activities with optional constraints may depend on availability of time, computation complexity requirements like time or additional value that may be derived to the business or customer.

During process execution, the constraint conditions become active and are evaluated by the process engines to guide further execution. Compliance is achieved when the constraint conditions are fulfilled i.e. the outcome of the process behaviour matches the prescribed behaviour. For instance, the flow and ordering of tasks matches their observed occurrence and positioning after execution.

Example 1: A policy specifies that activity D occurs before activity B and, between activities C and E. Any process model would be compliant if it conforms to the specified conditions i.e. activity D precedes B but occurs between C and E. Despite the specified ordering relations, implementation can benefit from the flexibility of constraint based modelling [117]. For instance, the rule is not specific as to whether B and E are parallel or sequential. In Figure 5.1, one way of expressing the order relations between activities A, B, C, D and E is illustrated using a BPMN model. The model captures the sequence of activities through ordered transitions such that the rule is complied with. Figure 5.1 represents a complex policy requiring composite control flow constraints be-

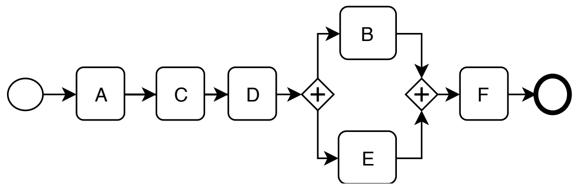


Figure 5.1: BPMN Model expressing ordering constraint

tween activities. Expression of constraints of this nature requires high flexibility offered by constraint based modelling paradigm [117]. Constraint based modelling offers the modeller freedom to represent models in various constraint satisfying ways. For example, variant 1 model in Figure 5.2 represents activities A and C modelled as exclusive activities. Non occurrence of activity B in the model does not affect occurrence of D. The constraint is still satisfied based on activities D, C and E.

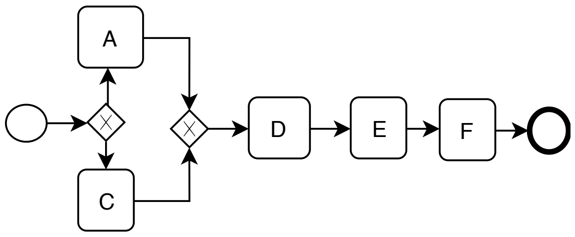


Figure 5.2: Variant 1 of the constraint model

Variant 2 in Figure 5.3 shows activities A and C modelled as exclusive activities. Activity D precedes B and it is between C and E thus conforming with the constraint.

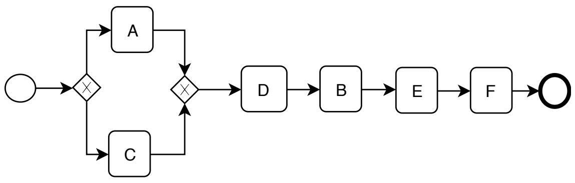


Figure 5.3: Variant 2 of the constrained model

Variant 3 in Figure 5.4 represents activities A and C as parallel while activities F and E as exclusive. Still the constraint is satisfied with D preceding B and between C and E.

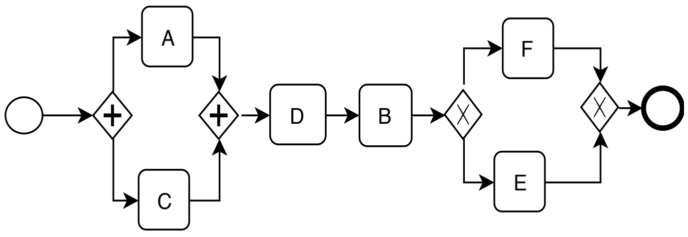


Figure 5.4: Variant 3 of the constrained model

However, any executions violating the constraint are not permissible; they exist when; activity D does not occur at all, when it occurs after activity B or when it occurs outside activities C and E.

5.3 Constraints for Collaborative Business Processes

Following the discussion from chapter 2 relating to the description and categorisation of constraints, it was emphasised that due to the characteristics of collaborative business processes, they are bound to comply with constraints from various external regulations. In this section, a formal structure of categories of constraints binding to collaborative business processes is summarised in Figure 5.5 based on constraints categorisations described in [159, 5, 6].

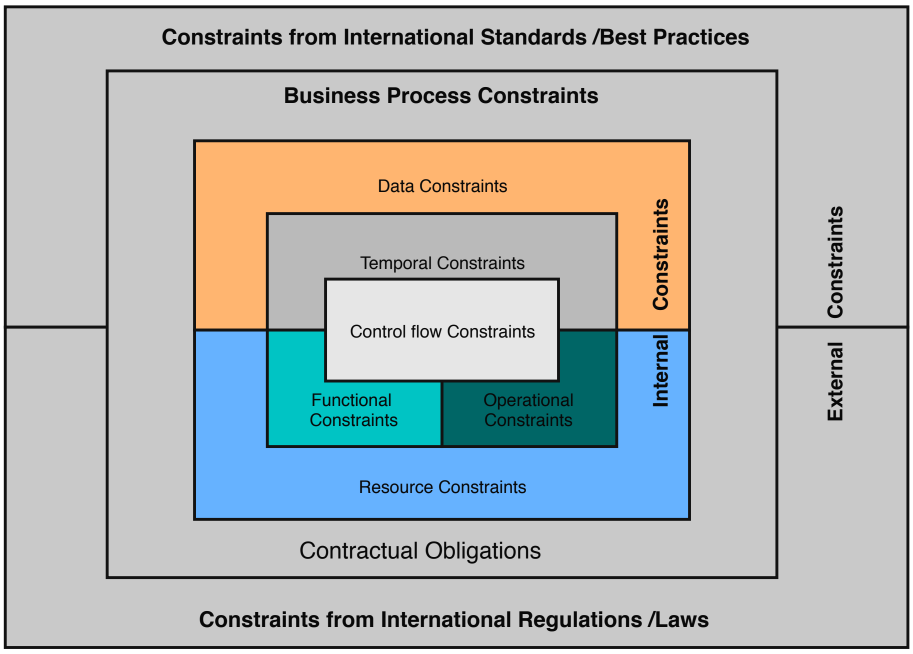


Figure 5.5: constraint categories based on structural perspectives of the business process

Furthermore, Figure 5.5 is a constraint relationship model illustrating logical relations between collaborative business process constraint categories. At the core of the model lies the control flow constraints which form a basis for all other internal constraints. It borders with the temporal, functional, operational, resource and data constraints which all form internal constraints. Beyond the internal constraints are contractual obligations; these integrate policies from partners. Next are constraints originating from the external regulatory agencies outside business environment. External constraints are regulations, standards, best practices and laws that to regulate the behaviour of the process beyond its borders or contractual obligations. Exemplified category based description of the constraints follows.

Temporal constraints and Examples

Table 5.1: Exemplified Temporal constraint patterns

Constraint	Description	Example
Instance duration	Instance duration	A set of activities from $a \ldots n$ must be executed within one (1) hour from the time execution starts. E.g. the processing of an order should last for one (1) hour from the time the order is submitted.
Delay	Period within which an activity can be delayed	Activity b will execute after exactly two (2) hours once execution of activity a is complete. E.g. after customer payment, shipment will be delayed for two (2) hours until payment is confirmed or reflected on the system.
Validity	Period within which an activity can be executed	Activity b will execute between 12:00 and 14:00 hours every day of the week. E.g. the shipment of goods takes place between 12:00 and 14:00. Or customer care service is available only during normal working hours.
Duration	The period for which an activity is scheduled to execute from start to completion	Activity a will execute for 45 minutes. Or activity a execution takes between 20 and 50 minutes.

Repetition	Period between which an activity can be repeated	After initial failed execution, activity a can be repeated for twice, otherwise it is restarted after 1 hour. E.g. log on can be tried for three (3) successive times, if it still fails it is restarted after one (1) hour.
Overlap	Period within which an activity can start and complete with reference to another activity's start and completion period.	Activity b is scheduled to start 30 minutes after activity a has started but can complete together, however b should not complete before $a$completes. E.g. pack items can start 3 minutes after verify order has started but cannot complete before verify order completes.

Table 5.2: Exemplified Control flow Constraint categories

Pattern /condition	**Description**	**Purpose for checking**	**Example**
Existence	An activity must occur in an instance or otherwise	Occurrence or absence of an activity	Activity C must exist in every instance of the process. E.g. every order must be verified.
Bounded Existence	An activity must occur for a specific number of times	Multiple occurrence of activities	Activity B executes several times until a required condition is fulfilled. E.g. the pick items activity is repeated until all items are picked.
Dependency	Execution of an activity based on occurrence or non-execution of another	Occurrence or absence of dependent activity	For activity C to occur, activity B and A must have executed successfully or otherwise. E.g. shipment of the goods depends on confirmation of payment.
Parallel	A set of activities must occur in parallel	Activities that are bound to occur in parallel	Activities C and D are mutually exclusive. E.g. upon order confirmation, invoice is sent to customer while the order is being processed.
Bounded Sequence	Number of times a chain of activities must occur	Number of occurrence of chained activities or otherwise	Activity B and C execute several times until required condition is fulfilled

Precedence	An activity must occur before another. This also true for chained precedence for limiting a chain of activities.	Order of occurrence i.e. activities that must occur before other(s)	Activities A and B are followed by C. E.g. every account balance checking is preceded by successful Login of the account holder
Response	An activity that must occur due to occurrence of another. This also true for chained response for limiting a chain of activities.	Order of occurrence i.e. activities that must occur after other(s)	Activity E will occur if activity C occurred. E.g. payment by cheque activates the cheque processing activity

Resource Constraints and Examples

Table 5.3: Resource Constraint categories with examples

Constraint	Description	Example
Segregation of duty	Requires separate execution of high risk tasks by different actors	Cheque processing is executed by two different actors.
Binding of Duty	Requires 2 or more related tasks to be executed by same resource.	The Doctor who diagnoses a patient must also prescribe drugs.
Delegation	Share or transfer permissions and associated responsibility from one actor to another.	Supervisor can delegate verify order to pickers.

Table 5.4: Data Constraint categories

Constraint	Description	Example
Data Visibility	Definition of data elements based on structure of the process and scope of accessibility of data	Task data; Describes data elements accessible by the task or by each of the components of the corresponding tasks blocks. Scope Data; Data elements defined which are accessible by a subset of the tasks in a case or defined according to several tasks that are coordinated. Multiple Instance Data; Tasks that occur multiple times in a case can define data specific to an individual execution instance Case Data; Data accessible by all components during execution of the case. Workflow Data; Data elements are accessible by all components in each case of the process and its context. Environment Data; Process components have access to external data

Data interaction i.e. internal & external	Definition of data elements based on how data is exchanged between process components and how their characteristics determine data flow	Data Interaction between - Task to task -Block Task to Sub Workflow decomposition -sub-Workflow Decomposition to Block Task -Multiple Instance Task
Data validity	Definition of data in a state that is useful and meaningful for task execution	Controls necessary to keep data up to date
Data availability	Definition of data in format that is ready for use and application	Defining which data has universal access and ensuring its universal availability
Data accessibility	Definition of data elements that make data accessible.	Access control and authorisation, regulation and legitimisation.

Data privacy	Definition of data elements that preserve privacy of data	Permissions: Represents a set of permissions granted to users to access data such as Permitted, Forbidden, permitted if condition is true. Permissions are linked to actions performed only if one has the permission to do so. Also, purpose is linked to permission; permission cannot be given unless a purpose is specified. It is also used to represent user consent. Conditions: Conditions that must be true to allow an action to be performed on data. Data Retention: Defines the period data is kept at the requester end.
Two – Three-Four-way matching	Requires values of two different data objects to match	Received goods must match payment invoice.
Authenticity	Requires Identification management for controlling data access	All users are identified and authenticated by the system.

Table 5.5: Temporal Constraints and Combinations with other Constraints

Temporal constraints	Delay	Validity	Duration	Repetition	Deadline
Control flow	Existence Precedence Response	Existence	Parallel	Dependency Bounded Existence Bounded Sequence	Existence Precedence Response
Resource flow	Authentication	Authentication	SoD Binding of Duty	Two – Three - Four-way matching	Authentication Privacy SoD BoD
Data flow	Data accessibility Data validity	Data availability Data accessibility Data availability Data visibility Data privacy	Data interaction	Data accessibility	Data accessibility Data availability Data visibility Data privacy.

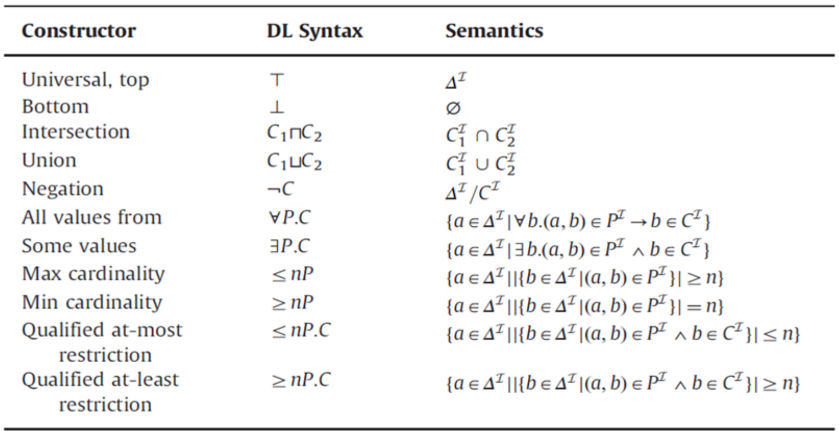
Tables 5.1 – 5.4 present set of general compliance patterns as observed from literature [45, 162, 46, 142]. For example, Table 5.1 presents temporal constraints described with examples namely Duration, instance duration, delay, validity among others. Table 5.2 presents exemplified control flow constraints patterns including existence, bounded existence, dependency, precedence, response inter alia. In Table 5.3, resource constraints are listed and exemplified including separation of duty, binding of duty and delegation. Lastly Table 5.4 describes data constraints with examples. Data constraints include data visiblity, data validity, availability and accessiblity, privacy among others. Because temporal constraints do not exist independently, Table 5.5 is a matrix matching temporal constraints with other constraints to benefit combination of constraints during their expression and specification. In subsequent sections, the constraint patterns are described formally by deriving logical relations using description Logic and LTL.

5.4 Constraint Expressions

Different forms of logic have been implored to define, express and specify constraints. For instance, studies by [117, 46, 60, 142] use different forms of logic that compose their proposed languages. However, these formalism remain difficult to comprehend by ordinary end users like compliance offices and other stakeholders who are the subject matter experts yet lack technical knowledge of defining and specifying constraints. In this thesis, application of descriptive logics (DL) is adopted and adapted as a less complex constraint expression formalism, upon which we base to compose a mechanism to express and specify constraints. The motivation to use DL is based on its rich syntactical and semantical vocabulary, which is yet easy to understand and use by ordinary users due to its closeness to natural language. Constraints expressed and specified in DL are easy for human intuition, understanding and interpretation. Besides, DL remains expressive enough to support reasoning over constraints and their eventual checking.

5.4.1 Description Logic

DL is a language used for formal representation of knowledge by facilitating formal expression and specification of requirements of knowledge base systems. DL extends into different types like spatial, temporal and fuzzy logics with different features to support various forms of expressivity and reasoning complexity. A DL diagram features concepts which represent sets or classes of a system, and role representations which establish relationships between concepts. Roles have value restrictions which impose constraints or limitations or upper bound or lower bound on the types and values that fill the role. Existential restrictions and value restrictions facilitate characterisation of concept relationships, while set theoretic notations are adopted like intersection, union and complement as concept conjunction, disjunction and negation respectively. The concepts under the domain of discourse are defined by characterising their relationships and properties with other concepts. It is not in this interest of the thesis to discuss full details of DL. However, we highlight concepts relevant for application to policy and regulatory requirements definition. In Figure 5.6, a set of DL applicable syntax and semantics are given.



Constructor	DL Syntax	Semantics
Universal, top	$\top$	$\Delta^{\mathcal{I}}$
Bottom	$\bot$	$\varnothing$
Intersection	$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
Union	$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
Negation	$\neg C$	$\Delta^{\mathcal{I}} / C^{\mathcal{I}}$
All values from	$\forall P.C$	$\{a \in \Delta^{\mathcal{I}} \mid \forall b.(a,b) \in P^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$
Some values	$\exists P.C$	$\{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in P^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$
Max cardinality	$\leq nP$	$\{a \in \Delta^{\mathcal{I}} \mid \lvert\{b \in \Delta^{\mathcal{I}} \mid (a,b) \in P^{\mathcal{I}}\}\rvert \geq n\}$
Min cardinality	$\geq nP$	$\{a \in \Delta^{\mathcal{I}} \mid \lvert\{b \in \Delta^{\mathcal{I}} \mid (a,b) \in P^{\mathcal{I}}\}\rvert = n\}$
Qualified at-most restriction	$\leq nP.C$	$\{a \in \Delta^{\mathcal{I}} \mid \lvert\{b \in \Delta^{\mathcal{I}} \mid (a,b) \in P^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\rvert \leq n\}$
Qualified at-least restriction	$\geq nP.C$	$\{a \in \Delta^{\mathcal{I}} \mid \lvert\{b \in \Delta^{\mathcal{I}} \mid (a,b) \in P^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\rvert \geq n\}$

Figure 5.6: DL Syntax and Semantics

The following section therefore presents and illustrates language application to derive formal constraint expressions.

5.4.2 Constraint Expression Mechanism - Application of Description Logic

In the adaptation of DL to our environment the business process is the domain of discourse while activities and constraints are concepts. Role representation is used to establish a link between the constraints and the activities while the role restrictions impose specific existential and value restrictions of a constraint over the activity. We use unary predicates to represent sets of individual constraints while binary predicates denote relationships between combined individual constraints, e.g. existence and response, i.e. an activity occurs in response of another activity that occurred. We further use Composite predicates to denote relationships between constraints from different pattern categories. E.g. The assignment of resources to execute an activity for a given duration. The combinations and adaptations yield a mapping illustrated with Figure 5.7.

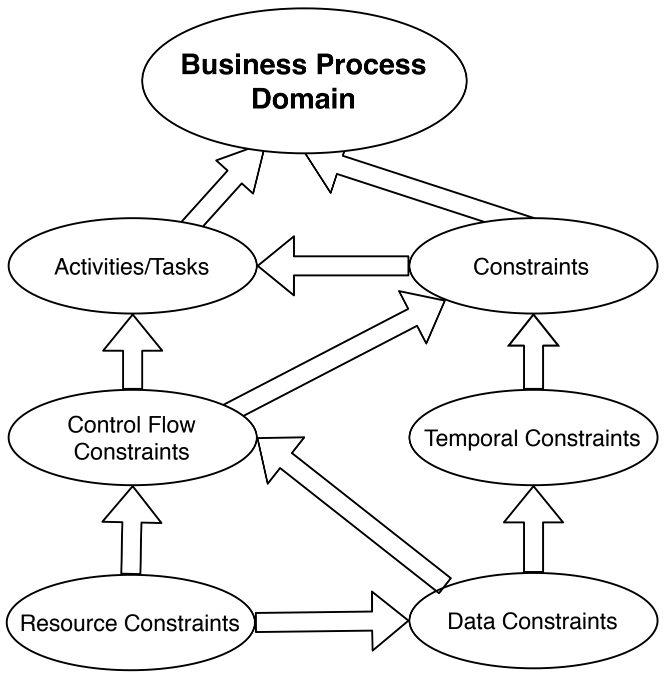


Figure 5.7: Modified DL Diagram

Based on this figure, it is possible to derive and express constraints, properties and relations among constraint. In this way, expressions are specified for control flow like response, existence, bounded existence, dependency and parallel constraints in Table 5.2 and resource constraints like Segregation of duty, binding of duty and delegation listed in Table 5.3, as well as data constraints like validity, access and availability, and authentication in Table 5.4 with temporal constraints in like duration, validity, repetition and overlap in Table 5.1. The sections that follow present the constraint expressions.

5.4.3 Control Flow Constraint Expressions

In this section, the presentation of constraints is categorised according to DL unary, binary and composite predicates. Unary predicates represent atomic constraints while binary and composite predicates represent combinations between constraints.

Unary Predicate Expressions for control flow Constraints

Control flow unary predicates are used to represent control flow-based constraints expressing ordering relations involving atomic activities or tasks. To fulfil the ordering relations, LTL operators and quantifiers are used for the purpose. The expressions are presented in Table 5.6.

Activity combinations are required to express relations between one or more activities. Such relations are represented by forming combinations between constraints using combinations of predicates known as binary predicates.

Binary Predicate Expressions for Control Flow Constraints

Two additional logical symbols are composed to achieve purposeful and meaningful expressions. These are; $\ll\leftarrow$ to represent 'precede', $\rightarrow\gg$ to represent 'leads to', $||$to represent parallel, and $\mapsto$ to represent dependence. Table 5.7 presents binary expressions specifying constraints defined with examples from use case 1.

Table 5.6: Unary Expressions for Control Flow Constraints

Pattern	DL Representation	Description	Example
Existence	Exist.Activity	Activity occurrence in a trace	$Exist.a$
	Exist.∀Activity	Every activity occurs in trace	Exist.$\forall(a, b, c)$
	Exist.∃Activity	Some activity occurs in trace	Exist.$\exists c$
	¬Exist.∃Activity	Some activity excluded from occurrence	¬ Exist.∃ a
Bounded Existence	BoundedExist.Activity	An activity can occur several times	BoundedExist.a
	BoundedExist.$^{k(n-1)}\forall$Activity	Every activity can occur for specific number of times	BoundedExist.$^{k(n-1)}\forall(a, b, c)$
	BoundedExist.$^{k(n-1)}\exists$Activity	Some activity occur for specific number of times	BoundedExist.$^{k(n-1)}\exists(a, b, c)$
	¬ BoundedExist.∃ Activity	Some activity cannot occur more than once	¬ BoundedExist.$\exists b$
Dependency	∃ Activity $\mapsto$ Depends Activity	Some activity depends on occurrence of another	$b \mapsto a$
	$\exists Activity$ ¬ Depends	Activity does not depend on another	$a\neg$ Depends or a_{init}
Parallel	∃ Activity $\rightarrow$ Parallel Activity	Some activity is parallel to another activity	$b\|\|c$

Bounded Sequence	$\exists$ Activity $\rightarrow$ BoundedSequence	A chain of activities occurs in sequence	$\exists(a, b, c...n$ $\rightarrow$ Bounded Sequence)
Precedence	Activity Precede Activity $\exists$ Activity $\rightarrow$ Precede Activity Activity $\neg$ Precede	An activity is preceded by another Some activity is preceded by another activity An activity has no preceding activity	$a \ll\leftarrow$ b $\exists a \ll\leftarrow b$ $a\neg \ll\leftarrow$ or a_{init}

Table 5.7: Exemplified Binary Expressions for Control Flow Constraints

Pattern	Requirement	DL Expression	Example	use case 1 Example
Existence	Non- occurrence of an activity leads to absence of another Non-occurrence of an activity causes a complement activity to occur. E.g. case of XOR	$\neg$ Exist.Activity $\rightarrow\gg$ $\neg$ Exist.Activity $\neg$ Exist Activity $\rightarrow\gg$ $\neg$ Exist.Activity	$\neg b \rightarrow\gg \neg c$ $\neg$ b $\rightarrow\gg$ $\neg$ b’	$\neg$Verify order $\rightarrow\gg$ $\neg$ Hand over $\neg$ Customer pickup $\rightarrow\gg$ $\neg$ Store delivery
Existence and Response	Activity occurs in response to occurrence of another	Exist.Activity $\sqcap$ Response.Activity	a $\sqcap$ $\rightarrow\gg$ b	Verify Order $\sqcap$ $\rightarrow\gg$ Hand Over
Existence and Precedence	Activity occurrence is preceded by occurrence of another	Exist.Activity $\sqcap$ Precede.Activity	a $\ll\leftarrow$ b	Pick items $\ll\leftarrow$ Verify Order

Bounded Existence and precedence	A chain of activities precedes an activity	BoundedExist.∃Activity ⊓Precede.Activity	$\exists(a,b,c) \ll\leftarrow d$	∃(Select order, Pack items)≪←Verify Order
	An activity precedes occurrence of a chain of activities	Precede.Activity⊓ BoundedExist.∃Activity	$(a) \leftarrow\ll \exists(b,c,d)$	∃(Verify order)←≪(Pack items, Hand Over)
Bounded Existence and Parallel	A chain of activities occurs in parallel to each other	BoundedExist.∃Activity⊓ Parallel.∃Activity	$\exists \sqcap (a,b,c) \rightarrow\Vert \exists(d,e,f)$	∃ (Pick Items, Verify Order) →‖ ∃ (Contact Customer,Change Item)
Bounded Existence and Dependency	Chain of activities occur depending on execution of another chain of activities	BoundedExist.∃Activity⊓ Depend.∃Activity	$\exists(d,b,c) \mapsto \exists(a,d,e)$	∃(Pack Items, Hand Over, Delivery)↦(Pick Items, Verify Order)

5.4.4 Resource Constraints Expressions

For simplicity and convenience, abbreviations are adopted for use in expressions representing resource constraints as follows: Available - Avail, Segregation of duty - SoD, binding of duty - BoD and Delegation - Del.

Unary Resource Constraint Expressions

Unary expressions representing resource constraints and their implications using description logic are as follows:

- Resource.SoD – Resources constrained with SoD constraint
- Resource.BoD – Resource is constrained with BoD constraint
- Resource.Del – A resource that can be delegated

Binary Resource Constraint Expressions

Binary combinations between resource constraints is possible under guiding principle that no combination between BoD and SoD for same activity executions at the same time. This comprises the access control restrictions, i.e. a resource cannot be constrained as BoD and SoD at the same time of allocation to an activity. This could otherwise result into deadlocks. Similarly, a resource cannot be available and unavailable at the same time. These restrictions must be observed at design time and verified to prevent violations that result into non compliance or deadlocks. Some of the realistic constraint combinations are;

- Resource.Avail $\sqcap$ SoD – Resource available for assignment as SoD.
- Resource.SoD $\sqcap$Del – Resources constrained with SoD and can be delegated.
- Resource.BoD $\sqcap$Del – Resource is constrained for BoD but can be delegated.
- Resource.Avail $\sqcap$ Validity [time] – Resource's availability is valid for a specific time.

5.4.5 Data Constraint Expressions

Unary Data Constraint Expressions

The section presents data constraints expressions based on Unary predicates using description language;

- Data.visible – Data items visible for each activity.
- Data.¬ visible – Data items not visible
- Data.interactive – Data items that can be interacted with.
- Data.valid - Valid data items
- Data.available - Available data items
- Data.¬available – Data items unavailable
- Data.accessible - All accessible data items
- Data.¬accessible – Data items inaccessible.
- Data.Privacy – Data items classified as private data
- Data.2-3-4WM – Data that requires matching to enable execution.
- Data.Authentication - Data items that require authentication.

Data constraints restrict the creation and access of the data by activity tasks and resources (roles and applications) over time.

5.5 Composite Predicate Constraint Expressions

This section presents composite predicate combinations involving all constraint categories to fulfil compliance requirements. They include the following;

5.5.1 Expressions Between Control Flow and Resource Constraints

Combination between control flow and resource constraints express conditions restricting assignment of resource actors to activities. The expressions specify activity behaviour in relation to their actors. The DL based expressions are represented as follows;

- Exist.Activity $\rightarrow$ Resource.Avail - The occurrence of an activity is assigned to an available resource actor. For example;
  $Ho \rightarrow Resource.[Avail]$ is a valid expression assigning any actor that will be available to execute the activity. Such activities assigned to any available actor are non critical or, they are already within the category of authorised actors.

- Exist.Activity $\rightarrow$ Resource.SoD - The occurrence of the activity is assigned to a resource actor constrained by separation of duty. For example;
  $Vo \rightarrow Verifier.[SoD]$ is a valid expression assigning the actor of role Verifier constrained by SoD to Verify order activity.

- Exist.Activity1$\sqcap$ Activity2 $\rightarrow$ Resource.BoD - The occurrence of the activity is assigned to a resource actor constrained by binding of duty. For example;
  $So \sqcap Pit \rightarrow Picker.[BoD]$ is a valid expression assigning the actor of role Picker constrained by BoD to execute Select order and Pick items activities. This implies that the actor executes both activities.

- Exist.Activity $\rightarrow$ Resource1.Del:Resource2 - The occurrence of the activity is assigned to a resource actor that can delegate to another actor. For example;
  $Po \rightarrow Packer.[Del] : Picker$ is a valid expression assigning the actor of role Packer who can delegate to actor Picker. Therefore, a given activity Ho can be delegated to several actors such that $Ho \rightarrow \bigsqcup[R_1, R_2, ...R_n]$ implies that Hand over order is assigned to R_1, R_2 and R_3

- Exist.$\forall$Activity $\rightarrow$ Resource.SoD - The occurrence of a set of activities is assigned to a set of resource actors constrained by separation of duty. For example;
  $Vo, Mo, CallCustomer \rightarrow Verifier, Supervisor, DutyManager.[SoD]$ is a valid expression assigning resource actors of role Verifier, Supervisor and Duty Man-

ager constrained by SoD to a set of activities Verify order, modify order and call customer.

- BoundedExist$^{(n+1)=k}$.Activity $\rightarrow$ Resource. Several occurrences of an activity is assigned to same resource. For example;
  BoundedExist.$^{(n+1)=k}$Pick Items $\rightarrow$ Picker. All the number of times within an instance of pick items are executed by same actor of picker role.

- BoundedExist$^{(n+1)=k}$.Activity$\rightarrow$Resource[SoD] – The number of times an activity can occur it is executed by a different resource activity is assigned to same resource actor. For example;
  BoundedExist.$^{(n+1)=k}$Verify order $\rightarrow$ (Verifier1 $\sqcap$ Verifier2).[SoD] - All the number of times the event instance of verify order is executed by a different actor (verifier1 or verifier2) of the assigned role (Verifier).

- BoundedExist.$^{(n+1)=k}\exists$Activity $\rightarrow$ $\sqcap\exists$Resources.[BoD] – Several occurrences of activities are assigned to same resource constrained as binding of duty for all occurrences.

- BoundedExist.$^{(n+1)=k}\exists$Activity $\rightarrow$ $\sqcap\exists$Resources.[SoD] – Several occurrences of activities are assigned to same resource constrained as separation of duty for all their event instances.

- BoundedSequence.$^{k}\exists$Activity $\rightarrow$ $\exists$ Resources.[SoD] - Activities occur as a chain for a number of times are assigned to different resources constrained by separation of duty.

- BoundedSequence.$^{k}\exists$Activity $\rightarrow$ $\exists$ Resources.[BoD] - Several activities occur as a chain for a number of times are assigned to a resource actor constrained by binding of duty.

Other expressions for control flow and resource constraints can be defined and specified following the same syntax and semantics. The above expressions are for illustration purposes.

5.5.2 Expressions for Data and Temporal Constraints Combinations

This section presents some of the expressions to illustrate composite predicate combination between data and temporal constraints. The relevant and applicable temporal constraint is the duration which is used to specify the period within which data can be available for access and use.

- visible.data $\sqcap$ [Duration] – Data items visible for a given duration
- interaction.data$\sqcap$ [Duration] – Data items that can be interacted with over a period of time.
- valid.Data$\sqcap$ [Duration] - Data that is valid for use for a period of time.
- Accessibility and Availability.Data $\sqcap$ [Duration] - Data that is accessible and available for all tasks and resources for specified duration.
- Data.Authentication $\sqcap$ [Duration] - Data accessible by authentication over given duration for tasks and resources.
- Data.Privacy $\sqcap$ [Duration] - Private data accessible through authorisation of tasks and resources for a specific duration.

More combinations of data and temporal constraints are possible following the illustrated expression mechanism.

5.5.3 Expressions for Control Flow, Resource, Data and Temporal Constraints

This sections presents predicate combinations for all constraint categories. The combinations represent means for complete constraint specifications that is close to natural language. This way, non expert end users can extract compliance requirements from source policy and regulatory documents and represent them as constraints to be complied with by the business processes.

- Exist.Activity $\rightarrow$ Resource.[SoD] $\sqcap$ Data.[available]$\sqcap$Time.[Duration] - The expression specifies that an activity is assigned to some resource constrained by separation of duty, and data access as available for a specific duration.

- Exist.Activity $\rightarrow$ Resource.[BoD] $\sqcap$ Data.[Private]$\sqcap$Time.[Duration]. The expression specifies that an activity will occur, assigned to a resource constrained as binding of duty, with data constrained with privacy for a duration of time.

- BoundedExist$^{(n+1)=k}$.Activity $\rightarrow$ Resource.[BoD] $\sqcap$ Data.[Authentication]$\sqcap$Time.[Duration] - The expression specifies an activity that will occur several times, with each time to be executed by a resource constrained by binding of duty, and to access data by authentication for a specific duration of time.

- BondedSequence.$^{k}\exists$Activity $\rightarrow$ Resource.[Del] $\sqcap$ Data.[Authentication]$\sqcap$Time.[Duration]- A set of activities to be executed for a number of times in sequence are assigned to resources which can delegate and share execution rights to other resources.

The expressions represented in the above sections do not cover all the constraints but only illustrate the mechanism to specify and express constraints in a formal less complex manner. Further illustrations are presented in subsequent chapters based on use case examples. However, while working with temporal constraint combinations it is important to note the categorisation in Table 5.5. These categories show the compound relations between temporal constraints and other constraint patterns and how the combinations are achieved. For example, availability and privacy as constraint examples from data constraints relate with the duration pattern implying that access to private data or availability of data are modelled and verified for a specific duration. The semantics adopted are intended to be as close as possible to natural language to achieve simplicity for non-expert end users like compliance officers. The syntax adopts use of logical quantifiers and operators from first order logic to achieve language expressiveness as well as support for reasoning about the model behaviour to achieve consistency and soundness.

5.6 Chapter Summary

In this chapter, the compliance constraint categories were listed with examples and represented as DL expressions. The chapter further introduced the DL language as a formal logic for constraint expression yet intuitive enough for easy understanding and application by ordinary users to formally gather requirements and constraints from their sources. This is so because of its closeness to natural language. The next chapter presents the formalisation and mapping of constraints into LTL.

Chapter 6

Formal Modelling of Compliance Constraints

6.1 Introduction

This chapter presents an approach for translation of policy and regulatory requirements into an interpretive format upon which enhanced reasoning can be applied to support compliance verification. A mechanism based on description logic and basic temporal logic syntax is presented using use case examples. The rest of the chapter is structured as follows. Section 6.2 presents the constraints definition procedure while in section 6.3 constraint specifications are expressed as formal logic. In section 6.4, formal definitions are given while in section 6.5 the validity and satisfiability of process behaviour is illustrated. Section 6.6 concludes the chapter.

6.2 Constraints Definition Procedure

Rules, policies and regulations are translated from ambiguous natural languages into formal constraints. These are then verified against business process models for compliance. Rules and policies specify what should be done in form of required behaviour or prohibited process behaviours by decribing conditions under which actions are permitted or forbidden [65]. These are interpreted by the process engine to automatically guide process execution. To achieve automatic interpretation, rules and policies must be well-formed and formalised. This way, misinterpretations and misunderstandings are prevented which would otherwise lead to constraint violations. A procedural approach is adopted in which control flow constraints are defined before other constraints. The underlying assumption is that a model must satisfy control flow constraints to before satisfying resource or data constraints. Activities must occur in the correct prescribed order to make it possible to check if they occurred at the right time or if they were executed by the assigned actor. Against this background, the subsequent sections present specifications of terms and concepts necessary for formalisation of constraints.

6.3 Representation of Constraints as Formal Logic

Following policy and regulatory specifications using description logic in section 5.4.2, formal definitions and specifications are presented in this section based on basic temporal logic constructs. Linear temporal logic (LTL) enables specifications of formulae expressing the future state of the system. In subsection 2.11.1, temporal logic was introduced as a formalism upon which various process verification tools are built. To formulate our formalisation logic based language to specify constraints, we borrow a set of operators from the LTL semantics and syntax. Some of the operators representing the language syntax are as follows [18];

If P and Q are path formulas representing events, then P, $P \vee Q$, $P \wedge Q$, XP, FP, GP, PUQ, PWQ and PRQ are path formulas of atomic propositions. The formulas are useful for expression of events in a trace upon which reasoning is applied. The combination of formulas follows the syntax and temporal operators given below and temporal formulas in Table 6.1 are used to express temporal requirements given f and g as temporal formulas.

$\sigma \models !f \Leftrightarrow (\sigma \not\models f)$

$\sigma \models fg \Leftrightarrow (\sigma \models f) \wedge (\sigma \models g)$

$\sigma \models f|g \Leftrightarrow (\sigma \models f) \vee (\sigma \models g)$

$\sigma \models f \rightarrow g \Leftrightarrow (\sigma \not\models f) \vee (\sigma \models g)$

$\sigma \models f \quad \mathrm{Xor} \quad g \Leftrightarrow ((\sigma \models f) \wedge (\sigma \not\models g) \vee (\sigma \not\models f) \wedge (\sigma \models g)$

$\sigma \models f <-> g \Leftrightarrow ((\sigma \models f) \wedge (\sigma \models g) \vee (\sigma \not\models f) \wedge (\sigma \not\models g)$

Table 6.1: Temporal Operators

Operator	**Syntax**
Next	Xf
Eventually	Ff
Always	$G f$
Strong Until	$f\, U\, g$
Weak Until	$f\, W\, g$
Weak Release	$f\, R\, g$
Strong Release	$f\, M\, g$

6.3.1 Constraint Specifications and LTL Definitions

To facilitate reasoning and analysis, the constraints previously specified (section 5.6) are translated into LTL to benefit from enhances reasoning. The translations show both atomic and composite constraints. Atomic constraints relate to a single constraint or representation of relation between single constraints category. In the previous chapter atomic constraints were represented as unary constraints. Table 6.2 presents the exemplified LTL constraint expressions based on DL control flow expressions earlier specified. Composite representations constitute combinations between constraints from different categories e.g. Control flow and resources. To compose the translations, the following definitions are considered.

Role Actor/ Activity Assignment

The concepts actors and users are used synonymously. Actors belong to roles whereas the roles are assigned to execute process activities as follows;

Role − Actor Assignment = UXR where R refers to roles with $r \in R$ and U refers to a set of actors with $u \in U$ such that (u, r) is a valid assignment of an actor to a role.

Activity - Actor assignment = A XU where A refers to a set of Activities with $ac \in A$ and U refers to actors with $u \in U$ such that *(u,ac)* is a valid assignment of an actor to an activity. Table 6.3 presents detailed formalised specifications of task and resource assignments exemplified with use case one.

Activity / Data Assignment

Activity execution requires access to data. For example, to execute the task 'order delivery' in use case 1, customer data in form of customer address should be accessible and available for this task. Further still, the type of actions that the user can do with the data must be pre-authorised i.e. action to read, write, modify or a combination of any or all. Therefore task data assignment is composed of a task type, data object (o), value v, and action (å).

Task data assignment TD= task type, data object value and action. The assignment is achieved by a function

$$f: ac \rightarrow o, v, \mathring{a}$$

which maps data and its attributes to a task to be executed by a subject.

Figure 6.1 illustrates task/ activity assignment and the required attributes for its execution. The assignment of data required for execution of delivery task is as follows;

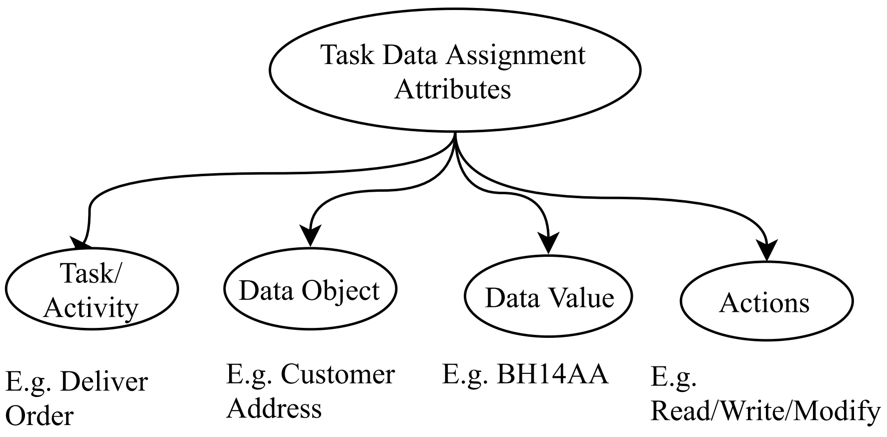


Figure 6.1: Task and data assignment attributes

- Task = Deliver order
- Data object = Customer addresses
- Data values = BH14AA
- Action = Read

Using DL, the expression below specifies task data assignment;

$$TD = ac \rightarrow (o \sqcap \nu, \sqcap \mathring{a})$$

$$TD = [DeliverOrder \rightarrow (Customer - address \sqcap BH14AA \sqcap Read)]$$

Using LTL, the expressions above are formalised as;

$$TD = a \rightarrow (o \wedge v \wedge \mathring{a})$$

$$TD = [DeliverOrder \rightarrow (Customer - address \wedge BH14AA \wedge Read)]$$

To formalise and enforce activity -data constraint assignments, Table 6.4 presents exemplified assignments, whereas Table 6.5 presents exemplified activity- temporal assignments. The two tables are composed of formal expressions for specifying constraints with examples from use case one.
Table 6.2 presents control flow attributes or elements based on LTL syntax and semantics listed in Table 6.1 that are useful for enforcing relations between control flow constraints to enable expressing of ordering relations that meet control flow requirements.

Table 6.2: Requirements and Constraint Formalisation

Constraint	Requirement	Formalised expression	Description	Applicable Example
Existence	Activity starts every process instance	$a_{init} \in \sigma$	Activity a is an initial activity for each instance	Select order is the starting activity
	Activity (a) exists in every trace	$G(a)$	Activity (a) Always occurs	Every order must be verified
	An activity can be executed many times	$GF(a)$	Activity occurs many times	Verify order is repeated execution as long as order details are not yet satisfied
	Activities can be skipped in the instance	$G(\neg a)$	Activity (a) will not occur	Contact Customer is skipped.
Until	Activity can occur for a specific number of k times	$G((a)F(a'^{(n=k)})$	Activity (a) loops until a condition is fulfilled	Pick items repeats until all items are picked Verify order is repeated until it passes satisfaction.
Precede	Activity occurs before another	$(a) \ll\Leftarrow (b)$	Activity (a) must execute for (b) to execute	Verify order precedes Hand over.
Next	An activity execution immediately follows after another	$(a)XF(b)$	Activity (b) must hold at the next state	Pack order follows Verify order.

Release	Activity (a) and (b) exist in the same instance	$(a)R(b)$	Activity (b) will be true until and including the point where (a) becomes true and remains true	-

Table 6.3: Formalisation of Activity and Resource assignments

Constraint	Requirement	Description	Formalised expression	Applicable Example
Assignment	Activity (a) is assigned to role actor r_1	$G(a, r_1)$	Throughout the model Activity r_1 executes (a)	G(Pick items,[Pickers])
Exclusion	Activity (a) will never be assigned to actor r_1	$G(F(a, [r_1]))$	Throughout the model Verify Order is never assigned to Pickers	G(F¬(Verify Order,[Pickers]))
Binding of Duty	Activities (a) and (b) are executed by same actor r_1	G($a \wedge b, [r_1]$)	Select Order and Pick items tasks are executed by Pickers	G(Select Order ∧ Pick items, [Pickers])
Separation of Duty	Activities (a) and (a') are executed by different actors r_1 and r_2	$G(F(a, [r_1]) \wedge (a', [r_2])$	Verify Order is executed by both Verifiers and supervisors	G(F(Verify order, $[r_1]$)∧(Verify Order', $[r_2]$)))
Repeated execution assignments	Activity (a) occurs several times each time executed by different actors	$F((a, [r_1]) \wedge ((a^n, [r_n])^k))$	Activity (a) is repeated with different actor for each execution	F(verify order,[verifier] ∧ ((Verify order')^n, $[Verifier'^n]$)^k)

Table 6.4: Formalisation of Activity - Data Assignments

Constraint	Description	Requirement	Formalised expression	Applicable Example
Accessibility and Availability	Activity (a) has data item (d) assigned whose value is (v)	Order item list is accessible and available for pick items and verify order activities	$G(a, r_1) \rightarrow (AA : (d.[value, Action])$	G(pick items ∧ verify order)→ ((order.[order list, Read])
Authentication	Activity (a) and its actor need authentication to access data item (d) with value (v)	Access to product order data requires authentication	G((a, r_1)→ (Authenticate: (d.[value, Action]= [True/False]))	G((Pick Items, [Picker]) → (Authenticate:(Order.[item list, Read]=[True]))
Privacy	Activity (a) and its actor need authorisation to access private data item (d) with value (v)	Access to customer data needs authorisation	G((a, r_1)→ (privacy: (d.[value, Action] = [True/False]))	G((delivery, [agent]) →(privacy: (customer data. [Address, Read] = [True]))

Table 6.5: Formalisation of Activity - Temporal assignments

Attribute	Description	Requirement	Formal Expression	Applicable Example
Within	Activity (a) occurs Within time duration k	Pick items is executed within one hour	$G(a) \rightarrow \mathrm{t}_{(\leq k)}$	G(Pick items)→ Duration$_{(\leq k)}$

Between	Activity (a) occurs between k time duration	Verify Order is executed between 20 and 30 minutes	$G(a) \rightarrow \mathrm{t}_{(k \wedge k)}$	G(Verify Order)$\rightarrow$ Duration$_{(20 \wedge 30) mins}$
After	Activity (b) will execute k_1 duration after execution of (a)	select order executes atleast 40 minutes after order submission.	$G(a).t \geq k \rightarrow b$	G(Order Submission).Duration$\geq$ 40 mins $\rightarrow$ Select order
Repetition Intervals	Activity (a) occurs several times after k duration between each occurrence	The duration of Delivery 1 hour but repeated in interval of 2 hours	$G((a) \rightarrow t_{(=k)} \wedge F(a'^{n}) \rightarrow t_{(=k)})$	$G((Delivery)$ $\rightarrow$ Duration$_{(=1hr)}$ $\wedge$ F(Delivery'^{n}$\rightarrow$ interval$_{(=2hrs)}$))

6.3.2 Activity /Task and Temporal Assignments

Temporal assignments to tasks are used to schedule time periods when the tasks are to be executed or when they are to occur. To achieve the assignment, temporal constraints are mapped with control flow constraints over activities. The temporal patterns are used to capture temporal rules and their instances in the business process executions and facilitate tracing for their compliance or identification of violations. Table 6.5 presents examples of control flow and temporal constraints formalised as LTL.

6.4 Formal Definitions and Expressions

This section presents a set of formal definitions and expressions for the required concepts necessary for later application to reason about constraint compliance and process behaviour.

6.4.1 Preliminary Definitions

Definition 6.4.1. Business Process (BP)

Business processes are made up of activities and relations between them. Where;

$BP = ac, R$ such that;

ac= A non-empty set of activities and $R = RXR$ are activity relations.

Activities are executed as events representing the different states of an activity at a given time e.g. when an activity event is triggered to start, it initiates and transits into execution state. When the event completes or is cancelled, the state changes to complete or fail respectively. For instance, select order activity from use case 1 can be represented in terms of event states; start, execute complete. Figure 6.2 illustrates the states an event can take whereas Table 6.6exemplifies the event states for select order activity.

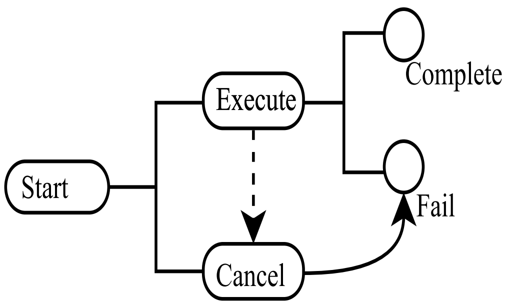


Figure 6.2: Activity event states

Table 6.6: Illustration of Event Status for Select Order Activity

Event	**State**	**Description**
Select order.Start	Started	Select order event started
Select order.Complete	Completed	Select order completed successfully
Select order.Fail	Failed	Select order failed

For convenience and simplicity during illustrations, Use case 1 activities are abstracted to short forms as follows; select order (So), pick item (Pit), verify order (Vo), pack order (Po), handle over (Ho) and Customer pick up or delivery (Cpd). Table 6.7 presents examples of possible event states for activities in the use case. The representation of failed event states is a useful pointer to the likely source of problem in case of a deadlock by looking out for specific events that failed during execution.

Table 6.7: Illustration of Event Status for Use Case 1

Activities	**Event states**		
	Start	Complete	Fail
Select order	So.Start	So.Complete	So.Fail
Pick Items	Pit.Start	Pit.Complete	Pit.Fail
Verify Order	Vo.Start	Vo.Complete	Vo.Fail
Pack Order	Po.Start	Po.Complete	Po.Fail
Hand over	Ho.Start	Ho.Complete	Ho.Fail
Customer Pick up or Delivery	Cpd.Start	Cpd.Complete	Cpd.Fail

Definition 6.4.2. Trace:

A trace, denoted as σ is the sequential occurrence of events. The traces are useful for checking process behaviour based on executions. In our approach simulation is used to generate sample traces to be used in constraints compliancy checking and verification. Related to the trace are the start and end events;

$e_{init} \subseteq \sigma$ = Subset of events which start a trace. It denotes activity events that must always start in an execution of a process instance.

$e_{end} \subseteq \sigma$ = Subset of events which end a trace. It denotes activity events that must always end in an execution of a process instance.

$n \in N$= The length of the trace and

$|n|$= Trace cardinality ranging from $0 - n^{th}$ event. E.g., $\sigma = (e_1, e_2, \ldots\ldots, e_n)$ is a complete trace.

$\{\}$= Represents an empty trace

$e_i = i^{th}$ Event in a trace

Definition 6.4.3. Process Instance (*Pi*):

Pi describes a set of events in prescribed order of execution. It may be formed of a combination of events from different traces whose execution shows accomplishment of a case. Events in a process instance are synonymous to logged behaviour describing occurrence of events during process execution. Table 6.8 presents examples of events occurring in different traces whose combination is based on some requirements (unspecified for now because they are not relevant). Each completed case represents a process instance. Examples given are for illustration purposes.

Table 6.8: Exemplified Events, Traces and Process Instances

Traces	**Events**					**Process Instance (Pi)**
	e1	e2	e3	e4	e5	
Trace 1	✓		✓		✓	Pi1
Trace 2			✓	✓	✓	Pi1
Trace 3	✓	✓			✓	Pi2
Trace 1			✓	✓	✓	Pi2
Trace 4		✓	✓		✓	Pi3
Trace 2	✓	✓	✓	✓	✓	Pi3

The table (6.8) shows different events from several traces making up process instances which in this case represent event occurrences.

Definition 6.4.4. Ordering relations:

Ordering relations describe the associations between process activities. Associations are indicators of the flow of operations and thus facilitate trace generation. An activity can occur before, during or after another activity or chain of activities determining its position. Similarly, with relation to timing constraints, an activity can be scheduled with a delay or specific period during which to occur. In all cases, the occurrence and position of an activity is determined relative to other activities and time. The occurrence and ordering of activities is specified following constraints. The ordering relations are further discussed with the control flow patterns in Chapter 4.

Definition 6.4.5. Constraints
Constraints impose restrictions over the behaviour of activities and thus determine how and when they execute. A constraint is a tuple

$$C = (c_t, \delta)$$

i.e. a set of conditions and rules that form constraints a process complies with. Such that $c \in C$ and $|c| = \phi$. Where:

c_t= Constraint type to denote a specific constraint as specified by the rule or policy e.g. occurrence of activity a.

δ = condition attributes specified by the policy as a requirement to be fulfilled. E.g. invoice amount $>$£ 600.

The constraint type represents constraints as per the categories discussed in Chapter 4 while the condition attribute specifies the specific data object and its value which is evaluated for each specified constraint.

Definition 6.4.6. Constraint Mapping to Activity
The mapping between constraints and activities is the assignment of the constraint to an activity. A function

$$f : ac \to C$$

is an assignment function where a constraint $c \in C$ is mapped on to an activity ac. E.g.

$$e_i = (a \to (SoD.Supervisor) \sqcap (Customerdata.[Address]) \sqcap (Duration.[6units]))$$

is an assignment of resource, data and temporal constraints to event e_i of activity *a*. The mapping involves events for activity *a* assigned to supervisors with separation of duty constraint granted access to customer addressed and to be executed for duration of 6 time units.

Further illustration; Based on use case 1, before an order is processed it is selected from pending orders which signals its change of status from pending orders to work in progress. After select order, items are picked. Constraints in this case are as follows;

$c_1 = (e_{init}) \rightarrow Selectorder)$ - A constraint assigning events for select order task as the initial event for every order processing instance.

$c_2 = ((PickItems) \rightarrow Response(Selectorder))$ - A constraint specifying the Response event Pick items after execution of select order.

Both constraints c_1 and c_2 illustrate control flow relations between activities in use case 1.

For convenience, $f_c(a)$ is used to refer to constraints that meet mapping requirements for activity *a* in order to achieve its compliancy at execution time. In other words, it refers to constraints that activity a should conform with. A collection of different activities and their constraints yields a constrained model.

Definition 6.4.7. Constrained Process Model

A constrained process model C_{pm} is composed of activities and integrated constraints. Any activity bound to a constraint is considered as a constrained activity. When a collection of constrained activities belong to a single model then such a model is referred to as a constrained model composed of a set of activities and their relative constraints.

$$C_{pm} = \bigcup ac \rightarrow C$$

C_{pm} is a union of all activitie s and assigned constraints. Constrained process model consequently leads to a compliance-aware process model in which the workflow Satisfiability problem can be partially solved i.e. where constraints assigned to the workflow activities meet the execution requirements specified in the business process policy and regulations to achieve a compliant business process.

Definition 6.4.8. Execution Behaviour

Behaviour is described from two fronts; i.e. prescribed behaviour and executed behaviour. Prescribed behaviour refers to behaviour defined at designed time while executed behaviour is one observed from process logs during or after runtime. The comparison between the two behaviours is an indicator of compliance or non-compliance. If both behaviours match on some aspects, then compliancy is achieved and the reverse is true for unmatched behaviour. Process execution behaviour for a constrained process model describes a set of process instances involving events of constrained activities. A process instance Pi expresses behaviour involving one or more events involving constraint attributes. A collection of several related process instances yields execution behaviour of the process for a particular case.

Let a set of process instances be $(Pi_1, ...Pi_n)$ with $Pi = e^i \rightarrow c^i$ as an activity event with assigned constraints. Process execution behaviour therefore involves several process instances representing various activity events and assigned constraints. Thus,

$$PB = e^1 \rightarrow c^1 \cup ... \cup e^n \rightarrow c^n$$

.

Illustration: The expression specifies process behaviour based on a process instance involving several events for various activities with constraints for resources, data and time. Some events are triggered by same users. Table 6.9 presents same information showing events for events for activities and constraints.

$PB = e_1 \rightarrow [((SoD.user1) \sqcap (CustomerAddress) \sqcap (Duration[6])), e_2 \rightarrow ((BoD.user2) \sqcap (CustomerOrderlist) \sqcap (delay[10])), e_3 \rightarrow ((SoD.user1) \sqcap (orderList)(Between[10]-[20])), e_4 \rightarrow ((user3) \sqcap (ProductList) \sqcap (Duration[15])), e_5 \rightarrow ((user4) \sqcap (ContactList) \sqcap (Duration[20])), e_6 \rightarrow ((SoD.user1) \sqcap (CustomerAddress) \sqcap (Duration[10]))]$

Table 6.9: Exemplified Process Instance Events for a Process Behaviour

Event	Activity	Constraints		
		Resource	Accessible Data	Time (units)
e1	Select order	SoD.user1	OrderList	Duration [6]
e2	Hand over order	BoD.user2	Customer orderlist	Delay [10]
e3	Select order	SoD.user1	Orderlist	Between [10]
e4	Pick items	User3	ProductList	Duration [15]
e5	Deliver order	User4	Contactlist	Duration [20]
e6	Select order	SoD.user1	Customer address	Duration [10]

Definition 6.4.9. Valid Process Behaviour
A process behaviour is valid if the execution process behaviour matches its prescribed process behaviour. In other words, the execution behaviour of the process complies with the specified policy and regulatory constraints. i.e. for every process instance, there exists a set of traces that exhibit the behaviour specified by the constraints.

$$V_{PB} = \forall Pi \in PB \exists \sigma | e_{i \leq 1 \leq j} \models f_c(a) \in C_{pm}$$

A process behaviour is valid if for every process instance, there exists a trace in which events or set of events meet the constraint requirements specified in the constrained model. We denote a set of all valid process behaviour as Ω.

Definition 6.4.10. Satisfiable Execution

Satisfiable process behaviour is the situation where for every process instance, all events or set of events in the traces meet the constraint requirements specified in the constrained model.

$V_{Sat} = \forall Pi \in PB \forall e_{i \leq 1 \leq j} \in \sigma \models f_c(a) \in C_{pm}$

A process behaviour is satisfiable if all events in the process instance match all events or set of events of the constraint requirements specified in the constrained model.

6.5 Application of Validity and Satisfiability of Process Behaviour

Process behaviour validity and satisfiability is an important for identifying potential process behaviour violations. It benefits end users to identify bottlenecks and violations involving activities, constraints and their executions early enough in the design as well as identification of design flaws to inform process design before actual process execution. For example; they facilitate identification of;

1. Necessary and potential constraints to be satisfied for a given execution behaviour, i.e. what are the key constraints required to fulfil a compliant execution behaviour given a set of activities?

2. Potential constraints that must be considered for execution to achieve compliant behaviour. Identification of such constraints helps to avoid violations during execution.

3. Potential activity events required for constraints satisfaction for compliant behaviour to be achieved, i.e. what activities must be executed to comply with requirements of specific constraints?

4. Potential constrained model activities necessary to achieve compliant behaviour satisfying key constraints

5. Critical constraints for a constrained model

The above indicators are useful for process designers as indicators for potential sources of violations or requirements to achieve compliancy by design. The section that follows presents expressions to achieve valid and satisfiable compliance indicators.

6.5.1 Expressions for Compliance Indicators

a. Indicator: Potential constraints for key activities e.g. potential constraints for activity

Purpose: to support identification of potential constraints required for an activity's execution to achieve compliant behaviour. Such constraints form necessary and sufficient conditions for the compliant process execution. The operation considers activity events as input and returns the required constraints that must be fulfilled by the execution of the activity.

$$P_{(a \to c)} = (c(V_{PB}) \models f_c(a)$$

The potential constraints for key activities are derived from a set of constraints required for valid process executions such that those constraints identified fulfil the execution requirements for activity a to be compliant.

b. Potential constraints fulfilling the conditions for execution of a compliant process behaviour

Purpose: to analyse, identify and return a set of potential constraints required for valid execution of a set of activity events in a constrained model.

$$P_{P_{B \to c}}=(\sum \forall c \in C \models (PB_{sat}))$$

The potential constraints to execute a compliant process behaviour are derived from a summation of all relevant constraints applicable for valid process behaviour such that the identified constraints are necessary to achieve a compliant process model.

c. Potential activity instances for execution to satisfy constraints

Purpose: To describe activities or their instances that must be part of every execution in order to achieve compliant process behaviour satisfying the constraint requirements. The operation takes in constraints and returns a set of activities necessary for execution to achieve conformance to specific constraints. This operation is useful is to provide *a priori* knowledge to the end users to ease identification of important constraints that must be complied with by the business process.

For example, every order processing instance must include events for verify order activity before the order is handed.

$$CA = (f_c(a) \in \sigma(Pi) \to PB_{sat})$$

The expression specifies activity *a* as a critical that must be part of instance execution to achieve compliant process behaviour. The operation identifies all constraints and relevant activities in process instances relevant for valid execution and satisfaction of constraints.

d. Critical activity instances for a constrained model Purpose: To identify those activity instances that must be executed or be part of a process execution if such execution is to be compliant with the requirements of the constrained execution model.
The critical activities are identified from the events in the process instances which are elements in the constrained process model not including non-critical events marked as primed events from primed process instances.

e. Critical constraints for a constrained model

Purpose: To identify a set of all critical constraints necessary for satisfaction by all process instance executions in order to achieve process compliancy. The operation considers a process model and all possible constraint assignments and returns the most critical constraints to be complied with.

$$c = (\forall c \in C \to PB | f_c(a) \in (Cpm))$$

The expression specifies all critical constraints necessary to achieve compliant process behaviour for a constrained business process model.

6.6 Chapter Summary

The chapter introduced constraints expression and specifications and their translation into a formal language. The outcome are formal constraints upon which reasoning can be applied to facilitate compliance verification.

Moreover, various definitions for concepts are provided which are necessary inputs for the next chapter (7). Lastly, the chapter introduced specifications for constraint validity and satisfiability for process behaviour and their application in identifying critical constraints that must be complied with by the process behaviour as well as critical activities that must be part of every execution instance to achieve constraints compliance.

Chapter 7

Compliance Verification Approach

7.1 Introduction

This chapter presents the overall compliance verification approach. As an integrated approach, the chapter presents the remainder of the components of the approach besides those discussed in the previous chapters. For example, compliance requirements elicitation and simulation based analysis were discussed and presented in chapter 4 while chapter 5 presented the logical expressions and formalisation of the compliance requirements. In chapter 6, the formal modelling of constraints is presented. Chapter 7 therefore concentrates on verification algorithms and how verification of collaborative business process models is achieved. The rest of the chapter is presented as follows; Section 7.2 introduces the overall compliance verification approach, In section 7.3, we show how to support verification of control flow constraints, resource constraints verification in section 7.4 and data constraints verification in section 7.5. In each section, the relevant compliance verification algorithms are presented. Section 7.6 presents the overall compliance verification algorithm while section 7.7 presents the process driven authorisation as an access control mechanism. Section 7.8 summarises the chapter with discussions.

7.2 Overall Compliance Verification Approach

Figure 7.1 presents an overall compliance verification approach showing three main steps. The first step is compliance constraints specification, the related formalisation techniques are explained in Chapter 4. The second step is compliance verification, the related algorithms are introduced in this Chapter. The third step is the feedback on the verification results. Section 7.2.1

further describes the steps.

7.2.1 Verification Steps

The first step: In this step the relevant rules and policies are extracted from source documents and compiled into a set of compliance requirements, defined to guide process behaviour . The set includes all requirements relevant for an organisation's business processes to comply with as sourced from all policies, contractual obligations and external regulations.

To support reasoning, model logic is used to translate the requirements into formal compliance constraints. In this case both Description logic and linear temporal logic are used. In section 6.3 of chapter 6, a mechanism for translating and formalising constraints is presented and illustrated in section 6.5.

The Second step: The business process model is verified for its compliance with formalised constraints. The goal is to check and ensure that the business process conforms with the required policies and regulations. Relatedly, in this step simulation analysis is used to illustrate the impact of change and variation in policy and regulations over the business process.

The Third step: The outcome of the verification forms the feedback reports displayed for users about compliancy or violation of the constraints. Outcome from simulation analysis shows the scenario reports and key performance indicators.

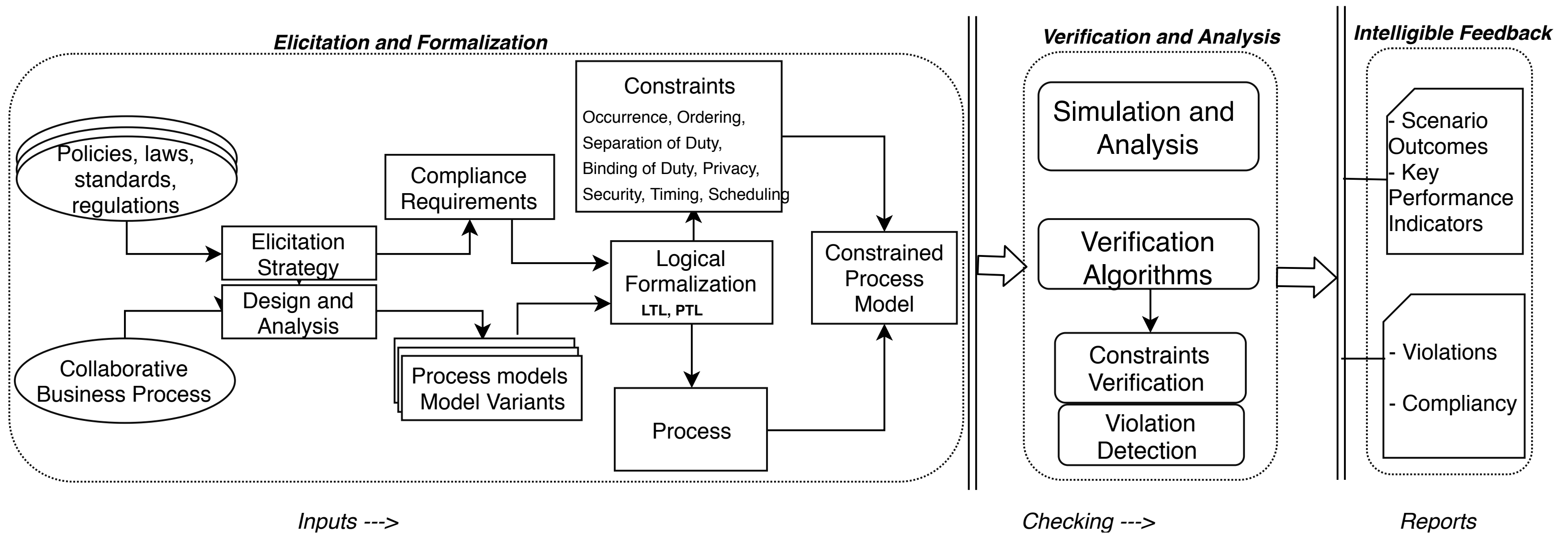

Elicitation and Formalization
Policies, laws, standards, regulations
Elicitation Strategy
Design and Analysis
Collaborative Business Process
Compliance Requirements
Process models Model Variants
Constraints
Occurrence, Ordering, Separation of Duty, Binding of Duty, Privacy, Security, Timing, Scheduling
Logical Formalization
LTL, PTL
Process
Constrained Process Model
Verification and Analysis
Simulation and Analysis
Verification Algorithms
Constraints Verification
Violation Detection
Intelligible Feedback
- Scenario Outcomes
- Key Performance Indicators
- Violations
- Compliancy
Inputs --->
Checking --->
Reports

Figure 7.1: Overall Compliance Verification Approach

The related work in chapter 2 shows the state of enormous academic and industrial work towards business process compliance verification in form of techniques, methods and tools. Notwithstanding, this thesis aimed at an abstract yet hybrid compliance approach to guide non-expert end users to verify collaborative business processes for compliance with policies and regulations through identification and detection of compliance violations. Achieving this goal required a set of artifacts iterative in nature, these are put forward in form of constraint specific, less complex and reusable compliance verification algorithms. Constraint specific in such a way that any given constraint can be checked by using a specific algorithm, and iterative in such a way that they can be applied at any stage in the business process life cycle. The next section describes how verification is supported.

7.2.2 Categories of Constraint Verification

The verification component of the compliance approach is formed of 2 types of checking i.e.

1. *The Simulation component*: Simulation is undertaken to generate traces to facilitate analysis and verification. the analysis involves predictive performance assessment of the business process based on variations in policy and regulations. Differing scenarios are generated and outcomes are analysed to support informed decision making. The details are discussed and presented in section 4.6 of Chapter 4.

2. *The Verification algorithm component*: This component is formed of algorithms that identify and detect compliance constraints violations. Various algorithms are composed for categorical constraint verification applicable in different ways, e.g. if a policy changes, users may want to check for compliance of existing processes with the changed policy. This way, only the relevant algorithm applies. An alternative is using the overall verification algorithm that combines all categories. Procedurally a business process is checked for compliance with all relevant constraints. This applies to new business process or those that have been modified significantly. In either case, the checking procedure in Figure 7.2 is followed. A business process is checked by detecting compliancy or violations to required behaviour expressed as constraints. Further, details of the checking are described in the algorithms presented in subsequent sections.

Figure 7.2 illustrates the compliancy verification procedure. The existing or new business processes are checked for conformance with defined constraints. If the process model is compliant,

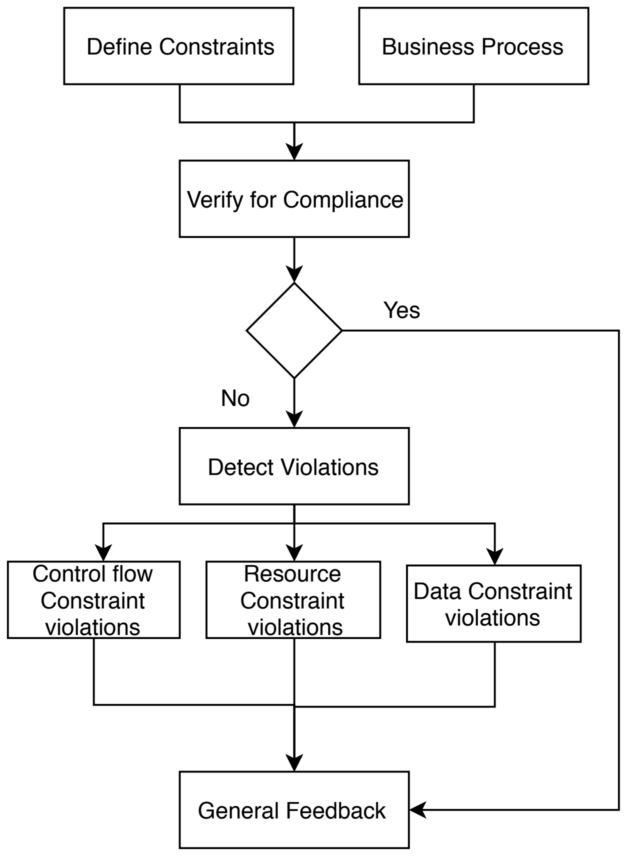


Figure 7.2: Compliance verification procedure

feedback is given, otherwise detection of non-compliant behaviour proceeds. Where the algorithms detect non-compliant behaviour, specific or general feedback is given about the violations according to the categories defined in chapter 2. To enable independent constraint checking, algorithms are composed according to same categories to permit constraint specific checking without need to follow a step wise procedure every time. The following section presents the algorithms according to their categories.

7.3 Control Flow Verification

The compliance verification algorithms that will be introduced later facilitate business process designers to check for the well-connectedness of the models to ensure that there are no errors like; 1) deadlocks, 2) improper termination, and 3) live locks. A well connected model facilitates checking for other system model properties like safety and liveness. Safety is a notion that nothing will go wrong in the model while liveness principle states that something good will happen.

This section presents the definitions and specifications for the functions that are used by the verification algorithms. The definitions follow the constraint categories.

7.3.1 Control flow Verification Requirements

Connectedness of the process model: Verification of how a process model is well connected is based on the modelling constructs like Sequence, AND, XOR and OR. It is important for the model to be well- formed from the design point of view even before other properties can be checked. This way, if a model's structural requirements are satisfied, then its soundness is consequently achieved [1, 161, 164]. At this level, verification targets to check how structurally well formed a model is in terms of sequence, parallelism, exclusive and inclusive choice constructs. In this section the structural requirements are defined and later we show how to verify for their conformance.

(a) Sequence: checking sequential connection between model objects. Based on definition 6.4.1 (business process) and definition 6.4.8 (Behaviour), a valid sequence is given by;

Sequence= $\sigma_i(a_1 + ... + a_n) \in Pi$

A sequence is a trace of activities from the initial to the n^{th} activity in a process instance satisfying a predefined order.

(b) Parallelism: checking connection between objects representing two or more tasks executed simultaneously and the possibility to converge at another object.

$AND = \sigma_i((a_1 - a_2) \wedge (a_1 - a_3)) \in Pi$

For a given trace in a process instance, any two interleaving tasks with no partial order relation conform to execution constraints if both tasks execute as per the constraint requirement.

(c) Exclusive choice: checking connection between objects representing disjoint tasks where one of them should execute.

$XOR = \sigma_i((a_1 - a_2) \vee (a_1 - a_3)) \in Pi$

For a given trace in process instance, any two disjoint tasks with no partial order relation conform to execution constraints if either of the tasks executes as per the constraint requirements.

(d) Inclusive choice: checking for connection between objects representing tasks where one or more alternative tasks can execute from a set of alternative paths.

$OR = \sigma_i((a_1 - a_2) \wedge (a_1 - a_3)) \wedge (a'_1 - a'_3)) \in Pi$

For a given trace in process instance, any two joint tasks with no partial order relation conform to execution constraints if one or of the tasks executes as per the constraint requirements.

Checking semantic consistency: Annotations are additional labels attached to model objects to represent constraints, data and additional artifacts as a way to provide more understandability of the model. Changes in policies affect model semantics. This may result semantic inconsistencies over similar models or model variants especially where there is lack of uniform applicable semantics [137] . Based on a collection model annotations over time, the goal of checking semantic consistency is to verify and improve model soundness, correctness and understandability to avoid model ambiguity, conflicts and inconsistencies. Figure 7.3 is an example of a BPMN process model annotated with data and conditional requirements.

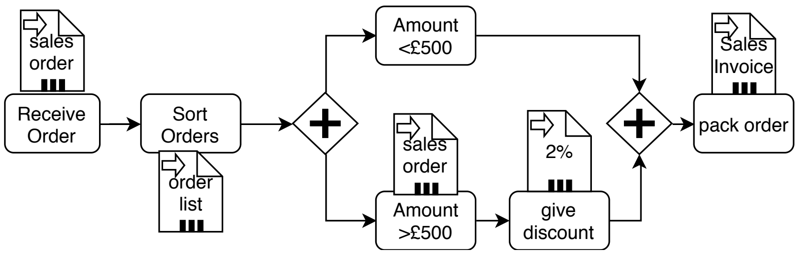


Figure 7.3: Annotated BPMN Model

Illustration:

Given 2 sets of semantics representing states S_1 and S_2 in a trace with literal sets such that S_1 = $\phi, \varpi, \delta, \alpha$ and S_2 = $\varphi, \neg\varpi, \infty, \vartheta, \varpi$ respectively. A consistent set should not include a member and its negation within the same set. Consider set S_2 with ϖ and its negation $\neg\varpi$. This yields a conflict which should not exist. Therefore, a set where members conflict results into an empty set. i.e. $\varpi \bigcup \neg\varpi = \Phi$ To update or create new states from existing ones e.g. change to state 3 s_3, a set of operations are involved. The subset involving a likely negation from each set is validated to a null and updated with a non-complementary subset to form a new state without conflicts. The scenario involving S_1 and S_2 to form S_3 would yield $S_3 = (\phi, \varphi, \varpi, \infty, \vartheta, \varpi, \delta, \alpha)$. Suppose the set is formed of events, it is alright for them to repeat if they are non-conflicting, e.g. . To achieve the combination would use a formula involving a concatenation. $S_1 + S_2 \longrightarrow S_3$ $S_3 = S_2(\subseteq (\varpi) \longrightarrow \Phi) + S_1$ Therefore: $S_3 = (\phi, \varphi, \varpi, \infty, \vartheta, \varpi, \delta, \alpha)$

The conflicting members set in S_2 are isolated into an empty subset while the non-empty set is concatenated with S_1 to achieve a consistent set S_3 i.e. without negating members. Checking for conflicts enables isolation and prevention of states that permit and at the same time prohibit event occurrence. This prevents deadlock and a live locks.

7.3.2 Specification of Control Flow Constraints

Control flow constraints include among others, existence and bounded existence, dependency, bounded sequence and precedence. Compliance to these constraints is verified in relation to temporal constraints to ensure that task ordering and occurrence follow time requirements. To facilitate the checking, we make the following definitions;

Specification for Existence (and Bounded Existence)

Existence constraint restricts an activity to occur in a specific order or time within a trace of a process instance. It also specifies ordering relations where specific activity events must start (e_{init}) or end (e_{end}) an instance. This way, the validity of an instance can be checked. To this effect definition 7.3.1 refers.

Definition 7.3.1. Existence (and Bounded Existence)

1. Existence for process instance validity.

   $Check.Exist : (e.ac = init) \sqcap (e.ac = end) \in \sigma$ Where: $e.ac$= event of an activity. The expression specifies a function to check initial and end activity events in a trace.

2. Existence of an activity within a process instance checked in reference to the control structures

   (a) If $(e.ac = AND)$ Return $\biguplus((a_1, a_2) \sqcap (a_1, a_3))$

   (b) If $(e.ac = XOR)$ Return $\uplus(a_1, a_2) \sqcup (a_1, a_3)$

   (c) If $(e.ac = OR)$ Return $\uplus(a_1, a_2) \sqcap (a_1, a_3) \sqcap (a_1, a_4)$

Application of the function

To illustrate the application of the function above, data in Table 7.1 is used to check the constraint requirements.

```
**for** each $\sigma \in Pi$ **do**
```

```
    $Check.Exist : (e.ac = init) \sqcap (e.ac = end)$
  **end for**
   **Return**
   $e.ac = init \notin seen$    /*Initial event is not in 'seen' events of the instance */
   $e.ac = end \in seen$    /* End event is in 'seen' events of the process instance. */
```

Using data populated in Table 7.1 with events, activities and process instances, we show the application of existence constraint specification and checking for its compliance or violation. Figure 7.4 shows resultant state graphs generated from the constraint checking of existence and bounded existence for all structural constructs (sequence, AND, exclusive and inclusive choices). The following verification requirements are addressed:

Requirement 1: All process instances start and end with activities a and z respectively.

Requirement 2: Between activities a and z, a set of other activities execute as part of the process instance.

Table 7.1: Sample events, activities and process instances

(a) Process Instances $P_1 - P_3$

Instances	Pi1				Pi2				Pi3					
Events	e1	e2	e3	e4	e5	e6	e7	e8	e9	e10	e11	e12	e13	e14
Activities	a	b	e	z	a	e	c	z	a	b	f	g	h	z
Time	2	4	3	5	2	3	6	5	2	4	6	4	8	4

(b) Process Instances $P_4 - P_5$

Instances	Pi4				Pi5		
Events	e15	e16	e17	e18	e19	e20	e21
Activities	a	i	m	z	a	z	m
Time	3	4	3	5	3	3	3

Requirements 1 and 2 in above section can be checked in the following way using the specified expressions.

```
  **for** $\sigma \in Pi$ **do** $Check.Exist : (e.ac = init) \sqcap (e.ac = end)$
     **Return**
    $init = a \forall Pi$    /*Returns activity *a* as initial activity for all process instances*/
    $end = z \forall Pi$ /*Returns activity z as end activity for all instances*/
end for
```

Based on the expressions, it follows that activity *a* is the initial activity for each process instance, so is activity *z* for end activity in each process instance. In terms of soundness, it shows compliance to termination is achieved by the possibility that each instance can start at *a* and end with *z*. However, the checking is not complete until we check for any possible violations of the behaviour.

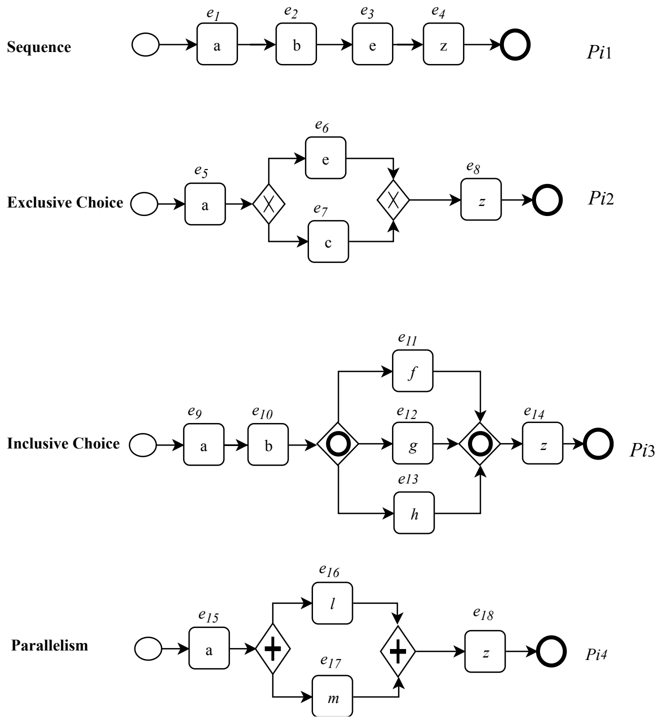


Figure 7.4: Resultant State graphs

Constraint Satisfaction Checking

we adopt to predicate functions for representing constraint satisfaction or violation.

- seen - Represents running activity events. If it is True that an activity event or set of activity events is in seen ($e.ac \in seen$), then the constraint is satisfied ($True \models C$). Otherwise it is violated ($True \not\models C$).
- finished - Represents executed activity. If it is True that an activity event or set of activity events is in finished ($e.ac \in finished$), then the constraint is satisfied ($True \models C$). Otherwise it is violated ($True \not\models C$) events.

Detecting violation to existence constraint

Violations to existence constraint are detected by checking for instances in which activities a and z are not initial and end activities respectively, and where the initial time assignments are not observed for all events. Circumstances leading to violation are checked from:

(a) Process instances where activity a is not the initial activity in a set of process executions, i.e. $a \notin seen$

From Table 7.1, it shows that events $(e15, 3, Pi4)$ partially satisfy the constraint since a is the initial activity for all instances. However, in terms of the temporal requirement the activity executes for longer time than scheduled, i.e. 3 units of time instead of 2 units.

(b) Process instances where activity z is not the end activity in all process executions, i.e. $z \notin finished$

From Table 7.1, it shows that trace $(e20, 5, Pi5)$ involves constraint violating event. Activity z is not the end activity for the constraint. There is a variance in execution duration where less than time is used 3 units are used compared to what was scheduled 5 units). This saves time as opposed to being a violation.

Specifications for Precedence and Dependence Constraints Verification

Precedence and dependence constraints are verified for activities whose existence has been confirmed. To verify that activity *b* is preceded by *a* and that the occurrence of b determines occurrence or non-occurrence of another activity *c*, we check for occurrence of b and return its preceding activity as well as the activity that occurs after its execution as its dependent activity, in other words activity *c* occurrence depends on activity b. The constraint is specified as the expression below;

Definition 7.3.2. Precedence and Dependence

$Check.Precede$ =$(a \ll\leftarrow b)$ /*checks for precedence of *a* over *b* */

$Check.Depend$ =$(c \mapsto b)$ /*checks for dependence *c* on *b**/

The expressions define activity *a* as a preceding activity to *b*, while occurrence of activity *c* is dependent on *b* such that *c* occurs if and only if *b* has occurred [167, 165]. The definition is used to specify constraint checking expression for the different control structures which are afterwards used in the algorithms. The checking involves;

(a) Checking if an activity has occurred in the trace $e.ac \in \sigma$.

Return error if $e.ac \notin \sigma$. Stop checking.

(b) Check for precedence and dependence constraints and returns outcome based on the routing constructs;

While $e.ac \in \sigma$ **do**

$((e.ac = a) \rightarrow Precedes(e.ac = b)) \wedge ((e.ac = c) \rightarrow Depends(e.ac = b))$: $(\exists c) \leftrightarrow (\exists b)$

Return $(e_{(i<=j)}) \in Pi$ /* Returns events satisfying or violating the constraints e.g. *c* occurs if and only if *b* occurs. Otherwise it is a violation*/

i. If AND /*output based on AND construct */

$$\left\{ \begin{array}{c} \cap^{e}_{i<=j} e.ac(aPrecedesb) \in seen = True \models C \\ \cap^{e}_{i<=j} e.ac(aPrecedesb) \notin seen = False \not\models C \end{array} \right\}$$

While verifying precedence constraint for activities based on AND construct, the checking returns a false if there are no seen events where activity *a* precedes activity *b*.

$$\left\{ \begin{array}{c} \cap^{e}_{i<=j} e.ac(cdependsb) \in seen = True \models C \\ \cap^{e}_{i<=j} e.ac(cdependsb) \notin seen = False \not\models C \end{array} \right\}$$

While verifying dependence constraint for activities based on AND construct, the checking returns a false if there are no seen events in which activity c depends on b

ii. If XOR construct */output based on XOR construct

$$\left\{ \begin{array}{c} \cup^{e}_{i<=j} e.ac(aprecedesb) \vee (aprecedesb') \in seen = True \models C \\ \cup^{e}_{i<=j} e.ac(aprecedesb) \vee (aprecedesb') \notin seen = False \not\models C \end{array} \right\}$$

*/Outcome for events satisfying or violating the precedence constraint on disjoint activities *b* and *b*' over activity *a*. A violation occurs when activity *a* is not seen among activities preceding activity *b* for all instances.

$$\left\{ \begin{array}{c} \cup^{e}_{i<=j} e.ac(cdependsb) \vee (c'dependsb) \in seen = True \models C \\ \cup^{e}_{i<=j} e.ac(cdependsb) \vee (c'dependsb) \notin seen = False \not\models C \end{array} \right\}$$

Set of events satisfying or violating the dependence constraint for disjoint activities c and c' over activity b. A violation occurs when activity *b* is not in seen activities where activities *c* and *c'* are seen among activities for the process instances.

iii. If OR /*Outcome based on OR construct*/

$$\begin{Bmatrix} \cup_{e+1}^{e} e.ac(aprecedesb) \wedge (aprecedesb'^{n}) \in seen = True \models C \\ \cup_{e+1}^{e} e.ac(aprecedesb) \wedge (aprecedesb'^{n}) \notin seen = False \not\models C \end{Bmatrix}$$

The occurrence of activity *b* is preceded by activity *a* where more than one alternative paths are permissible. If events of activity *a* are in seen and finished, then the precedence constraint is satisfied. Otherwise it is violated.

$$\begin{Bmatrix} \cup_{e+1}^{e} e.ac(adependsb'^{n}) \wedge (adependsb'^{n}) \in seen = True \models C \\ \cup_{e+1}^{e} e.ac(cdependsb) \wedge (c'^{n}dependsb) \notin seen = False \not\models C \end{Bmatrix}$$

The occurrence of activity *b* is preceded by activity *a*. If events of activity *a* are in seen and finished occurring before activity *a*, then the dependence constraint between *a* and *b* for all alternative paths is satisfied. Otherwise it is violated.

iv. If Sequence: constraint checking based on sequence construct is checked in the same way as specified expressions illustrated above.

Definition 7.3.3. Other control flow constraints

The illustration involved the definition and specification of existence, bounded existence, precedence and dependence constraints. However, Other control flow constraints like Response, bounded response inter alia can be extended into definitions and specifications in the same way as illustrated in sections above. For time and space limitations not all control flow constraints are specified. After the definitions and specification of constraints and checking functions, control flow compliance checking algorithms are composed.

7.3.3 Control Flow Verification Algorithms

Based on the above discussions, specifications and function definitions, a set of control flow based algorithms are composed to check compliance of the business process with control flow constraints. To make the algorithms self-contained and independent the definitions below are used for all algorithms. The general assumption is that events are ordered in a total order over time.

Predicate Functions

- Business process: =BP
- Process Instances: $Pi = \{\sigma_i, ..., \sigma_n\}$
- Trace(σ): Logical activity events.
- Events in a trace = started, seen, finished where;
  - started = $\{\}$ $-$ Set of started activity events.
  - seen = $\{\}$ $-$ Set of seen or running activity events.
  - finished = $\{\}$ $-$ Set of finished activity events.
- e.ac: Activity Events

Verifying for Basic Process Instance Validity

Sub-algorithm 1 checks for the basic validity of the model based on activity events that start and end a process instance. The algorithm checks for activity events designated to start or end a process instance. If start events are not in a set of 'started ' events ($e.ac \notin started$), it implies the activity has not started. If it is not in 'seen' activities ($e.ac \notin seen$), or 'finished' ($e.ac \notin finished$), it implies that the activity is not in execution or not completed. The same principle applies for the end activity events. In this case a violation is reported for activities not started, not in seen and not in finished.

Verifying for Compliance with Existence constraint

The existence constraint refers to constraints that restrict the occurrence behaviour of an activity. The algorithm verifies for occurrence of activity events in a process instance as per required behaviour specified by the policies governing operations. The events are fully ordered by time. It is intended to address the following verification requirements;

Requirement 2.1: Check for activities scheduled to occur but never start.

Requirement 2.2: Detect deadlocks by checking activities that start but never complete execution.

Based on algorithm 2, violation of the existence constraint is detected if any of the event activity states is not among the events that are started, executing or completed within the seen and finished event sets.

Algorithm 1 Basic Process Instance validity

```
1: $Input$:
        a. All $Pi$
        b. Constraints
2: for each $Pi \in BP$ do
3:     $e.ac = e.init, e.end$
4:     if $e.ac = e.init \notin started, seen, finished$ then
5:         ”Violation of validity for initial activity event”
6:     end if
7:     if $e.ac = e.end \notin started, seen, finished$ then
8:         ”Violation of validity for initial activity event”
9:     end if
10: end for
11: No violation of basic process instance for the given business process
```

Verifying for Compliance with Precedence constraint

Precedence constraints restrict the ordering relations between activities based on occurrence of a previous activity. In collaborative business processes characterised by multi-party executions, checking the precedence of activities benefits transparency in partner responsibility by knowing which activities must occur before others and who should execute them. In case of deadlocks, it is possible to point to the source of the problem. To facilitate verification of compliance with precedence constraints for activities, algorithm 3 is composed and presented addressing the following requirements;

Requirement 3.1: Detect activities that are potential sources of precedence violation.

Requirement 3.2: Use compliant behaviour to determine any likely violations based on the routing constructs

The algorithm checks precedence condition activity event over an action event. Violation occurs where the condition does not lead to the action or where the action occurs without the condition activity. For example activity a_1 is the precedence condition for occurrence of activity a_2. The occurrence of a_2 before occurrence of a_1 is a precedence constraint violation that algorithm 3 identifies.

Algorithm 2 Existence Constraint Checking

```
1: $Input$ :
       a. All $Pi$
       b. Constraints
2: for each $Pi \in BP$ do
3:     $e.ac.State \in Started, Executed, Completed$
4:     if $e.ac.State \notin Started, Executed, Completed$ then
5:         ”Violation: Existence constraint violated. Activity never occured”
6:     end if
7: end for
8: No violation of existence constraint for the given business process
```

Verifying for Compliance with Response constraint

Response constraint restricts execution of activities based on evaluation of a condition on the current activity. The activity will then execute in response to the outcome of that condition e.g. If a cheque is approved, then it can be issued. Issue cheque is a response activity from approve cheque. Execution issues arise if the condition is not evaluated or evaluates falsely leading to deadlocks or live locks. Algorithm 4 in this section checks for compliancy with response constraint over a set of activities. The following verification requirements are addressed:

Requirement 3.1: Detect activities likely to lead to response-based violations. *Requirement 3.2*: Detect deadlocks resulting from non-responsive activities.

Algorithm 4 checks for Response constraint between activity events where an activity condition (e.ac.Condition) responds to an action activity event (e.ac.Action) where, occurrence of the action activity in the seen and finished events not as a response from the conditional activity event violates the response constraint.

7.4 Resource Compliance Verification

Verification for compliance with resource constraints aims at checking for the fulfilment of the resource requirements by the business process such that no violations exist in its behaviour.

Algorithm 3 Precedence Constraint Checking

```
1: $Input$ :
        a. All $Pi$
        b. Constraints (Precedence)
2: for $Pi \in BP$ with Precedence constraints C do
3:     $Prec = e.ac.Condition \Rightarrow e.ac.Action$
4:     if $(Prec \notin seen, finished)$ then
5:         Violation (“Precedence constraint violated”)
6:     end if
7: end for
8: No violation of Precedence event in the business process
```

7.4.1 Specification of Resource Constraints

This section specifies the resource constraints as formal expressions and functions applicable in the resource verification algorithms to detect violations. The constraints are separation of duty, binding of duty and delegation.

- Separation of duty: Requires two disjoint activities (a_1, a_2) to be executed by different resource actors (r_1, r_2). Such assignment is based on preliminary specification for actor (user) and task assignment as defined in section 6.3.1; In light of the above, SoD specification for r_1, r_2 over (a_1, a_2)) is defined as;

  Definition 7.4.1. SoD

  $\nexists r_1 \in U : ((a_1, a_2), (r_1)) \in RP$

  The assignment of SoD constraint serves as a guard preventing a single actor in a role from executing two disjoint activities. It follows therefore that there should not exist any assignment of an actor r_1 to execute both activities (a_1) and (a_2) in a user task assignment. The contrary is a constraint violation.

- Binding of duty: BoD requires two tasks (a_1, a_2) to be executed by the same resource actor (r_1). BoD verification checks to ensure compliance to this requirement the contrary of which is a violation. Following preliminary definitions above, specification for activities (a_1) and (a_2) as BoD i.e. BoD(a_1, a_2) is given by the definition;

Algorithm 4 Response Constraint Checking

```
1: $Input$ :
       a. All $Pi$
       b. Constraints
2: for all $Pi \in BP$ with Response constraints C do
3:     Response = e.ac.Action $\Rightarrow$ e.ac.Condition
4:     if ($Response \notin seen, finished$) then
5:         Violation:    "Response constraint violated"
6:     else if e.ac.Action $\Rightarrow$ e.ac.Condition $\neq$ Response $\in seen, finished$ then
7:         "Violation, condition activity occurred without induced action activity".
8:     end if
9: end for
10: No Violation of response constraint on the provided business process instances.
```

Definition 7.4.2. BoD

$r_1 \in RP : \forall((a_1, a_2), r_1) \in \text{RP}$

For each actor assignment involving activities (a_1) and (a_2), one actor should be assigned for their execution. Contrary to the assignment is a constraint violation.

- Delegation: For tasks designated to specific resource actors, delegation enables sharing of execution rights with other actors. Two scenarios result where; the delegator shares and retains execution rights to the object or completely delegates and retains no execution rights to the delegate. Delegation is a practice in business operations to ensure business continuity. It also guards against activity dead locks that result from over constrained resources that create time lags and delays, or improper implementation of constraints like the four-eye principle.

  Specification of the delegation constraint requires information about subjects (users who delegate and those delegated to), and objects. Therefore, given two (2) users r_1 and r_2 where r_1 delegates activity *a* to r_2, the expression below specifies the delegation constraint;

Definition 7.4.3. Delegation

$(a, r_1) \in UT | r_1 \rightarrow Delegate(a, r_2) : (a, r_1 \wedge r_2)$

User (r_1) with rights to activity *a* delegates rights to user r_2 but retains execution rights such that both users are now assigned to activity *a*. $(a, r_1) \in UT | r_1 \rightarrow Delegate(a, r_2)$

Similarly, the above specification indicates that User (r_1) with rights to activity *a* delegates to (r_2) by passing on all the execution rights such that the delegator can no longer execute the activity.

7.4.2 Definitions for Resource Constraints

To facilitate the checking of compliancy to resource constraints the following definitions are relevant. Given a trace $\sigma \in (a_1, a_2, a_3)$ and a set of two users r_1 and r_2 of instance Pi_1, the following functional definitions are employed by the algorithm during resource constraints compliance verification.

While σ $\in$(a$_1, a_2, a_3), (r_1, r_2)$ =Pi$_1$ do

Check.SoD =$((a_1, r_1) \wedge (a_2, r_2))$ /*checks compliance to user assignment over activities a_1 and a_2 based on SoD constraint*/

Check.BoD =$((a_1, a_2), r_1)$ /*checks compliance to actor assignment over activities a_1 and a_2 based on SoD constraint */

Check.Delegate =$(a, r_1 \wedge r_2)$/* checks compliance to delegation constraint for activity *a* between actors r_1 and r_2 */

Return is used to generate the outcome from compliance checking showing whether compliance or violation is achieved based on the different structural controls i.e. AND, Parallelism, OR and XOR.

7.4.3 Resource Compliance Verification Algorithms

The resource verification algorithms apply the specifications and definitions in previous section to check process behaviour. The previous definitions are applicable for algorithm 5;

Algorithm for SoD Constraint Verification

Verifying for this constraint involves checking traces of the process instances to ensure compliancy to its requirement. The SoD algorithm is composed for this purpose. Where non-compliant behaviour is detected the algorithm returns a violation. The following verification requirements are addressed;

Requirement 4.1: Identify and detect resource assignment violations that lead to role conflicts based on SoD.

Requirement 4.2: Identify and detect roles and tasks upon which SoD violations are likely to occur.

Algorithm 5 SoD Constraint Verification

```
1: $Input$:
        a. All All $Pi$
        b. Constraints (SoD)
2: for all actors (r) where C= SoD do
3:     $(r_1, r_2)$.SoD $\rightarrow (a_1) = (a_1, r_1), (a'_1, r_2) \in e.ac$
4:     if $(e.ac) \in seen, finished \neq \quad ((r_1, r_2).SoD)$ then
5:         Violation: SoD constraint violated for $r_1$ and $r_2$ over $(e.ac)$
6:     end if
7: end for
8: Return No violation of SoD constraint for the provided processes.
```

while running, algorithm 5 checks for all users constrained by the SoD constraint SoD(user) and are assigned to a set of activities. The execution of activities (e.ac) by the constrained resource actors must observe the SoD constraint requirements. The activity events of (c.ac) should exhibit the behaviour to satisfy the constraint. On contrary, if the activity events in the process instances are not the same as the activities described in the behaviour, then the SoD constraint is violated. The behaviour is not seen (SoD user is missing). Otherwise no violation if the same user executed activity event $e.ac$.

Algorithm for BoD Constraint Verification

Verifying for BoD constraint involves checking the traces in the process instances to ensure compliance with its requirements by the business process. A BoD checking algorithm is composed to detect non- compliant behaviour. The following verification requirements are addressed by the algorithm;

Requirement 5.1: Identify and detect resource assignment violations that may lead to role conflicts based on BoD.

Requirement 5.2: Identify and detect roles and tasks upon which BoD violations are likely to occur to prevent deadlocks.

Algorithm 6 BoD Compliance Verification

```
1: $Input$:
       a. All $Pi$
       b. Constraints (BoD)
2: for all actors (r) with C= BoD do
3:     $(r_1)$.BoD $\rightarrow (a_1, a_2) = (a_1, r_1), (a_2, r_1) \in Pi$
4:     if $(e.ac) \in seen, finished \neq$ $(r_1)$.BoD then
5:         Violation: ”BoD constraint violated for $r_1$ over $(e.ac)$”
6:     end if
7: end for
8: Return No violation of BoD constraint for the provided processes.
```

Similar to SoD, if the constraint assigned as part of the activity, the events of that activity should exhibit the behaviour to satisfy the constraint. If the behaviour is not seen (constrained user is missing) then the constraint is violated. Otherwise no violation if the same user executes the assigned activities.

Algorithm for Delegation Constraint Verification

For a role to delegate to another it must have exclusive rights to the activity. Verifying for delegation constraint involves checking the traces in the process instances to ensure that all delegated actors have assumed their responsibilities to prevent task and resource redundancy where resources or tasks become idle, or deadlocks resulting from no resources assigned to execute tasks. A delegation checking algorithm is composed to check non-compliant behaviour. The following verification requirements are addressed by the algorithm;

Requirement 6.1: Verifying that all delegated roles assume their execution responsibilities.

Requirement 6.2: checking for violations likely to lead to role conflicts or idle roles as well as permission leakages.

Delegated users become valid users to execute activities not initially assigned. If a delegated user is not part of the valid user set, or if such users are not the ones that executed the running activities or finished activity set, then the delegation constraint is violated.

Algorithm 7 Delegate Compliance Verification

```
1: $Input$:
       a. All $Pi$
       b. Constraints (Delegate)
2: for all actors (r) where $C = Delegate$ do
3:     $(a_1, r_1)$.Delegate $(r_2) = (a_1, r_1 \wedge r_2) \in e.ac \rightarrow r_2 \in$ Valid Users
4:     if $(e.ac) \in seen, finished \neq \quad ((r_1, r_2)$.Delegate) then
5:         Violation: Delegate constraint violated for $r_1$ and $r_2$ over $(a)$
6:     else if $r_2 \notin$ Valid Users then
7:         Violation: ”Delegated role not existing”
8:     end if
9: end for
10: Return No violation of Delegate constraint for the provided Business processes
```

7.5 Data Compliance Verification

Verification of compliance with data constraints checks for how a model conforms with data requirements. Such requirements include; data availability and accessibility, Authentication and Privacy. Other requirements forming data constraints include; visibility, interaction and validity security requirements [127, 128]. For convenient checking and verification enforcement, the different patterns are compounded into the sub categories discussed below;

1. Data availability and accessibility (AA) constraints: Besides exclusive access requirements, data should be available and accessible to a basic level to facilitate work progress. Besides, data should be available and accessible whenever required. Verification of AA constraint requires checking for compliance with availability and accessibility data requirements.

2. Data Privacy constraint: the requirement to observe privacy of data justifies the establishment of access control and authorisation. Privacy constraint originates from the GDPR data privacy principle where organisations are required to build data privacy as part of their systems. verifying for data privacy involves checking for enforcement of privacy controls over data.

3. Authentication constraint: Authentication is a constraint to achieve basic security of data and systems by requiring users to be identified and given access. Authentication involves

the process of validating the identity of a registered user before allowing access to the protected resource. As a data constraint, authentication restricts access to data by requiring prior user login and profile authentication. It is based on identity management where digital identities are managed based on organisational security policies to ensure that only necessary and relevant data is shared using user identity and profile data as well as data governance functions.

Similar to privacy, compliancy to security constraint is demanded by many regulatory standards like GDPR and Anti money laundering. Specifically, GDPR emphasises security by design. Integrating security constraints and checking for their compliance in the process model is therefore important to meet policy and regulatory requirements.

7.5.1 Specifications for Data constraints

Boolean conditions are used to evaluate data access conditions are true or false. Depending on the outcome, access is granted or denied. If a trace is true to the conditions specified, then it satisfies the constraint. Otherwise it is false and violates the constraint. To that effect, the following specifications and definitions are useful for the data checking algorithm. Given a set of activities a_1, a_2 and a_3, assigned to resource actor (r_1) and requires access to product catalogue data (Pcd). Access to this data is constrained by access and availability, i.e. only 'Read' action can be granted. If the assignment is true according to the executed behaviour, then the trace (σ) satisfies $(\models)$ the constraint.

Definition 7.5.1. Accessibility and Availability (AA)

$\sigma \in (((a_1, a_2, a_3), r_1) : (Pcd.[Read]) : AA)$

If $(\sigma = True)$ then $\sigma \models AA$

The definition specifies accessibility and availability constraints for Pcd data object with action read granted to r_1 for execution of activities a_1, a_2 and a_3. During verification, the data compliance verification algorithm checks for compliance to the constraint for the data object, action by the user and tasks. If the outcome shows that the trace is true to the constraint requirement, then the trace satisfies the availability and accessibility constraint. Otherwise, its a violation detected for the AA constraint.

Definition 7.5.2. Authentication

$\sigma \in (((a_1, a_2, a_3), r_1), (Pcd.[True/False]) : Authentication)$

If $(\sigma = True)$ then $\sigma \models Authentication$

The definition specifies access control by authentication granted for accessing Pcd data with actions to read and write for role actor (r_1) who executes activities a_1, a_2 and a_3. Satisfaction of the authentication constraint is achieved if the trace of the executed events show exhibit the specified behaviour. Otherwise, a violation is detected for the authentication constraint.

Definition 7.5.3. Privacy (Prv)

$\sigma \in (((a_1, a_2, a_3), r_1), (Pcd.[Read]) : Prv)$

If $(\sigma = True)$ then $\sigma \models Prv$

The definition specifies Privacy constraint for accessing Pcd data where action to read private data is to be granted to the resource actor r_1 who executes activities a_1, a_2 and a_3. During verification, the privacy compliance verification algorithm checks the constraint for its satisfaction before access can be granted to read private data. If the trace is true for the specification, then the constraint is satisfied and thus compliance achieved. Otherwise, it is a violation detected for the privacy constraint.

7.5.2 Algorithms for Verifying Compliance with Data Constraints

For independent checking, algorithms 8 to 10 are composed for each constraint based on predefined specifications and definitions in section 7.5.1. The algorithms also consider function definitions in previous sections.

Algorithm for Access and Availability Constraint Verification

Verifying for data access and availability Constraints ensures that basic non-exclusive data is accessible and available with less restriction to enable accomplishment of basic tasks. algorithm 8 is composed to the effect. Violation occurs if role actors or tasks are denied access to data constrained by AA or where the permitted action type differs from the initial assignment, e.g. modify action type instead of read action type. The verification requirements addressed by algorithm 8 are;

Requirement 7.1: Ensure that required data is available and accessible for all tasks and role actors as required by AA constraint. This prevents events from executing without access to data. This prevents deadlocks where running events have no access to data or data is not available and events keep waiting for it.

Requirement 7.2: Identify and detect AA constraint violations likely to lead into data access denial.

Algorithm 8 Access and Availability Compliance Verification

```
1: $Input$:
        a. All $Pi$
        b. Constraints (AA)
2: for all data with constraint $C = (AA : [Read/Write/Modify])$ for actors (r) do
3:     Assign = $(r, e.ac) \rightarrow$ AA:Data Item.[Read/Write/Modify]$\equiv$ True
4:     if (Assign $\in seen, finished \not\equiv$ True ) then
5:         Assign $\not\models$ AA
6:         Violation: "Deadlock due to denied access to data. AA constraint violated"
7:     end if
8: end for
9: Return No violation of AA constraint for the provided processes if $Assign \in seen, finished \models AA$
```

Violation of AA constraint as per algorithm 8 exists when tasks or their actors $(r, e.ac)$ are denied access to data whose constraint is AA. This violation leads to a deadlock or livelock. Deadlock occurs if running activities are denied access to data necessary for the process to continue in execution. Whereas, the livelock occurs when a task is denied access to data stays in waiting mode stagnating process execution. The other form of violation may occur when the activity finishes execution without necessary data. This leads to wrong outcomes which do not comply with specifications.

Algorithm for Verifying Compliancy with Authentication Constraint

Authentication verification algorithm 9 verifies for compliance by checking that role actor credentials match the credentials stored in a database of authorised actors as well as the database for access privileges over tasks. The algorithm checks for three forms of Authentication errors which are the sources of authentication related violations:

- Access leakage which occurs when non-authenticated users gain access to data.
- Deadlocks which occur when users are authorised to execute activities but access to data is denied for technical or logical reasons e.g. improper configurations.
- Authentication breach which occurs when non-authenticated activities or users intention-

ally gain access to data. This is traced from running or finished events.

The following verification requirements are addressed by the algorithm;

Requirement 8.1: Prevent security lapses or leakages by checking actor identify and detect unauthenticated access to data by task executors or roles.

Requirement 8.2:. Detect authentication violations upon tasks based on access types.

Algorithm 9 Authenticity Data Constraint Checking

```
1:  $Input$:
        a. All $Pi$
        b. Constraints (Authenticity)
2:  for all data where C.Auth = Data item.$[Permit/Deny]$ do
        Assign =$r, e.ac \rightarrow$ Auth:Data Item.[Permit]$\equiv True$
3:      if (Assign $\in seen, finished \not\equiv$ True ) then
4:          Assign $\not\models$ Auth
5:          Violation: "authenticated access denied to restricted data."
6:          if $\exists$ actor $r_n \in$ Assign: Auth$\equiv False$ then
7:              Violation: "Access leakage, non-authenticated actor $r_n$ accesses data. "
8:          end if
9:      end if
10: end for
11: Return No violation of Authenticity constraint for the provided business process.
```

Actors are permitted or denied access to data by authentication. Where data constrained by authenticity is accessed by non-authenticated actors, it implies access leakage i.e. data is accessed by actors without authentication.

Similarly, where access to data is is denied to authentic actors, it leads to a deadlock since they cannot execute the current work in progress.

Authenticity compliancy checking algorithm checks for permitted or denied access to restricted data based on actor identities and roles. Where the assignment to data does not match the prescribed access policies, a violation is detected. Similarly, violations are identified from traces where transactions have occurred if the assignment does not match the traces ($Assign \not\models Auth$).

Algorithm for Verifying Compliancy with Privacy Constraint

Privacy constraint is enforced by means of access control and authorisation. Authorisation involves the process of validating that the authenticated user is granted permission to access the requested resources. Privacy as a data constraint restricts access to data regarded private as defined by GDPR. Data that is not available to the public is accessible by fulfilling authorisation requirement. Violation to privacy constraint is checked targeting two forms of errors; deadlocks and privacy breach.

- Deadlocks occur when the executing events authorised to access data are denied access for technical or logical reasons e.g. improper configurations,
- Breach to privacy i.e. non-authorised activities eventually access private data and execute.

To verify for these errors in a business process, algorithm 10 is composed. Authorised actors are granted permission to Read/Write/Modify private data items. Therefore compliant traces or transactions are those where the Assignment is equivalent to the authorised actions ($Assign \equiv Authorise$). Violations are detected or identified in traces where authorised permissions differ from the assigned ($Assign \not\equiv Authorise$).

Algorithm 10 Privacy Data Constraint Checking

```
1:  $Input$:
        a. All $Pi$
        b. Constraints (Privacy)
2:  for all data where C=Privacy:[Read/Write/Modify] for actors $(r)$ do
3:      Assign= $(r, e.ac) \rightarrow$ Privacy: Data Item.[Read/Write/Modify] $\equiv$ Authorise
4:      if $(Assign \in seen, finished) \not\equiv$ Authorise then
5:          Assign $\not\models$ Privacy
6:          Violation: ”Authorised actors denied access to private data”
7:          if $r_n \notin (r, e.ac) | r_n \in$ Authorise then
8:              Violation: ”Access leakage, non authorised actor access to private data”
9:          end if
10:     end if
11: end for
12: Return no violation of Privacy constraint for the processes if $Assign \models Privacy$
```

The other form of violation is where privacy constrained data exists outside the restricted boundary. This leads to a leakage since it is accessible by non-authorised actors. Similarly, where authorised data is not visible in 'seen' and 'finished' events it signifies a violation in form of a deadlock where data was not available or accessible to facilitate task execution. Authentication and privacy constraints are enforced by means of process driven access control and authorisation (PDAC) [84]. Section 7.7 discusses the PDAC concept in detail.

7.6 Overall Compliance Verification Algorithm

The overall compliance verification algorithm is a general algorithm that integrates the specific constraint checking algorithms into a single algorithm to check the entire business process behaviour.
The application of this algorithm is two fold:

- It can be applied to verify a business process where a large amount of modifications have been made necessitating checking the entire model for constraints compliancy, or
- Where a business process is designed from scratch automatically requiring full scale verification for compliance with policy and regulatory requirements.

Algorithm 11 Overall Compliance Constraint Verification Algorithm

```
1: $Input$:
       a. All $Pi$
       b. All Constraints
2: for all $Pi$:C = Control flow, Resource, Data, and Temporal do
3:     Verify compliance with control flow constraints
       Trace validity $\rightarrow$ Call algorithm 1
       Existence $\rightarrow$ Call algorithm 2
       Precedence $\rightarrow$ Call algorithm 3
       Response $\rightarrow$ Call algorithm 4
4:     if $Pi \models C = True$ then
5:         Verify compliance with Resource constraints
           Check SoD $\rightarrow$ call algorithm 5
           Check BoD $\rightarrow$ call algorithm 6
           Check Delegate $\rightarrow$ call algorithm 7
6:         Verify compliance with Data constraints
           Check AA $\rightarrow$ call algorithm 8
           Check Auth $\rightarrow$ call algorithm 9
           Check Privacy $\rightarrow$ call algorithm 10
7:         Message = Compliance status for Control flow, Resource, Data constraints
8:         Return overall compliance feedback for the provided business process.
```

7.7 Process Driven Access Control and Authorisation

PDAC is a concept we proposed in [86, 84] as a mechanism towards realisation of an automated and agile, yet less complex solution to overcome the challenges of non-compliance to security and privacy constraints. The motivation and rationale was based on the compliancy demands of the 2018 revised GDPR. At the dawn of the May 2018 launch of the revised GDPR version, big companies like Facebook, Inc [136] and Google LLC [133] were already faulted for data privacy breach. The GDPR articles of interest to this thesis are the principles of security by design and privacy by design. The former principle requires security of the data to be built within the information system design. The latter principle requires transparency from the data protector and

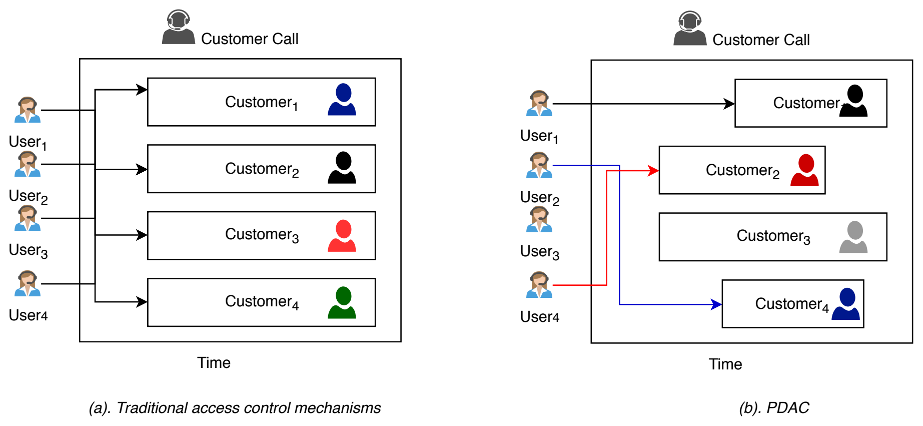


Figure 7.5: Illustration of PDAC vs Traditional access control mechanisms

processor to make known to the data owner the status of their data i.e. when it is being collected, processed and transmitted. Before collection and processing, the data owner's consent must be sought.

PDAC leverages existing solutions to enhance access control and authorisations to achieve automated compliancy, especially with dynamic policies and regulations. It ensures regulated and legalised data access based on its need to accomplish a specific process instance. As a divergent access control mechanism the from existing access control mechanisms, access under PDAC is based on the entire process instance by assessing the purpose, time and instance as opposed to the subject, object or action to be committed. This is a paradigm shift from the traditional access control models based on tasks [147], roles [147, 134, 50] and attributes [82, 79, 78] which grant and authorise more access than what is required. This violates the data privacy principle.

Despite their role in security and privacy administration, classical access control mechanisms are unable to support modelling and enforcement of security and privacy requirements presented by current workflows which must as well comply with many other regulations. Relatedly, workflows supporting collaborative business processes present more complex and dynamic security and privacy requirements that require agility to implement which is not provided in the current mechanisms. They grant roles more authority and permissions beyond what may be required. Figure 7.5 part (*a*) illustrates authorised users in a call centre granted full access to all customer records indiscriminately. They have access to records all time. Part (*b*) illustrates PDAC where users are granted access to a single record per session of time a customer is being served.

Various extensions to the classical access control mechanisms have been suggested. In Table 7.2, a summarised description of mechanism extension is presented together with PDAC. It is noticeable that the most common constraints dealt with are SOD and BoD. The suggested PDAC mechanism differs from the classical ones to address privacy and authentication constraints.

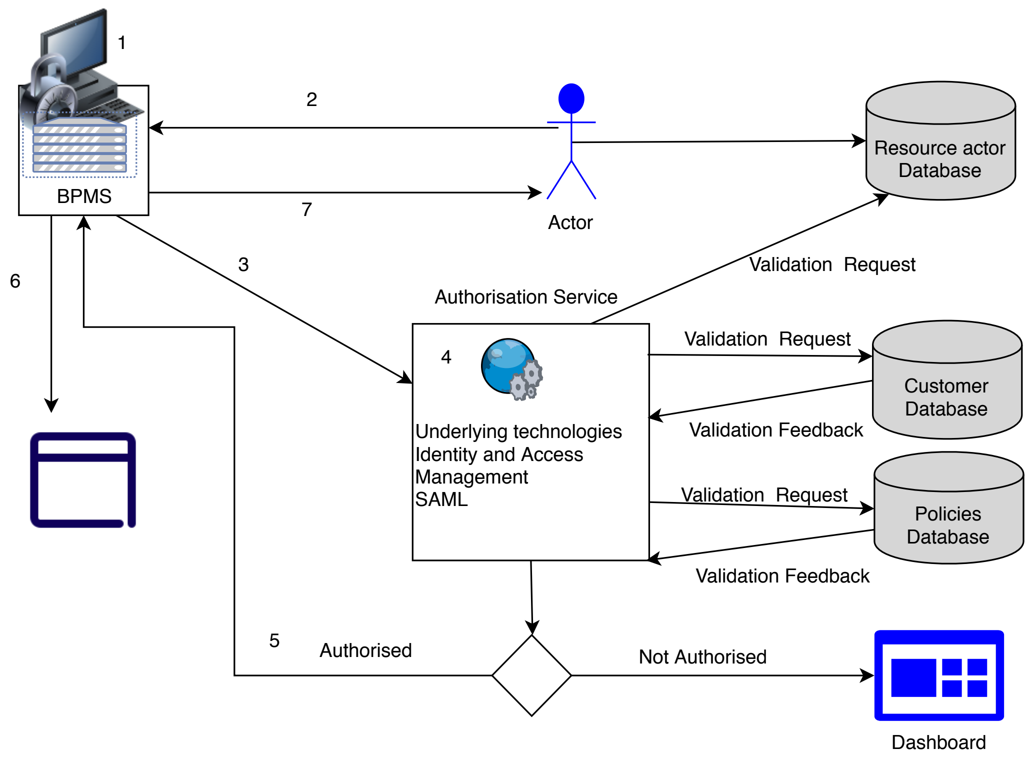
Table 7.2: Research on extensions of Access control mechanisms

Proposal	**Constraints**	**Mechanism**	**Output**	**State**
Support dynamic assignment of access controls based on the task instance context and task states	DSOD, BOD, Temporal constraints	BAC and RBAC	AC agent enforcement architecture	Design time, Runtime
Support modelling of constrained workflows for local and global constraints such that a sound workflow constrained schema exists where authorised user can execute a complete workflow instance	SOD, BOD, cardinality constraints	TBAC and RBAC	Formalised constrained sound workflow	Design time
The management of authorisations of organisation roles in a process view	SoD, duty of conflict	TBAC and RBAC	Algorithm	Design time
Authorisation and Access control model for giving subject access to objects during task execution	No concern for SoD or BoD	RBAC	Authorisation and access control Model	Runtime
A privacy-aware BP modelling framework supporting reasoning and enforcement of privacy constraints	Separation of tasks, Binding of Tasks, Necessity to know	User Roles	Extension of BPMN 2.0 to PrVBPMN	Design time
PDAC - Support process driven access control and authorisation	**Privacy, authentication and security constraints**	**Process Instance, Time**	**Compliance verification Algorithm**	**Hybrid**

7.7.1 Implementation architecture for Process Driven Access Control and Authorisation

Access to data is granted by authorisation and revoked automatically in two ways i.e. i) Once the purpose for which access was granted is accomplished, and ii) When the assigned duration expires. In either case, the resource actor ceases to have access to data. For example, in Figure 7.5 user is assigned access to a single customer's data for an instance of a call and access will cease the moment the call ends.

During execution, when access to data is required, the authorisation service is invoked to check the assigned access privileges. It then provides feedback for granted or denied access and provide message to the user via the dashboard.



1. Activity started
2. User accepts tasks
3. BPMS work list handlers issues data authorisation token
4. Authorisation engine validates request token with policy and customer databases
5. Token validated and issued to BPMS
6. The token is stored in the browser/ user client
7. Actor executes activity

Figure 7.6: PDAC Authorisation Service Architecture

Within the business process management system an activity event is initiated as step (1) shows. The activity is then assigned to a resource actor which will accept it in step two (2). The activity now exists in the work list of the actor (system user) in the BPMS. The BPMS issues an authorisation token request to access the required data in step (3). In step (4) the authorisation service is managed by the authorisation engine implemented by underlying technologies like identity and access management (IAM) and Security Assertion Markup Language (SAML).The authorisation involves validation of the request against user identities, policies and customer data in their specific databases. A collection and validation of a combination of these parameters legitimises access authorisation. The token is validated either offline with a short duration session token or with digital signature online validation. In step (5) a validated token is returned to the BPMS authorising activity execution by the actor and stored in the browser or user client profile in steps (6) and (7).

7.7.2 User Authentication

SAML technology supports enforcement of user identification and authentication. The user signs into the client portal e.g. a browser which sends an authentication request to the user identity database. The database authenticates the user by generating SAML authentication assertions that identify the users and their information.

The browser contacts the validation service with the SAML assertion which requests temporary security credentials and creates session for sign in. The sign in is sent to the browser granting access to the users based on policies in the policy database.

7.7.3 GDPR Implementation

The customer self-service point is for implementation and fulfilment of GDPR requirements. Enforcing compliance to GDPR requirements is achieved by enabling;

- Data owners have access to personal data by means of automated access.
- Restrict processing of data by data owners by directly interacting with data processors.
- Data modification and deletion through a self service interface.
- Data portability to enable data transfer serviced by the data owner.
- Audit and monitoring of data by its owner at any point in time.

7.8 Summary and Discussions

This chapter has presented a set of algorithms as category constraint specific algorithms as part of artifact contribution from this thesis. Before the algorithms, the chapter has shown how to define and express specifications for checking constraints and eventually composed algorithms that apply them for compliance verification. For each constraints category, an algorithm is composed and presented. Algorithms are presented as independent and self-contained algorithms to facilitate different but specific constraint checking. In whole when combined, overall model verification is accomplished.

For each category of constraints, an overall algorithm can be composed. An example of such is algorithm 12 that combines all algorithms. For each algorithm, inputs are defined, which are processed to return intelligible outcome showing constraint compliance or violations. The intention of algorithms is to benefit cases where one is interested in checking a model for compliance to a particular change in policy. The requirements in this case do not have to necessarily check the entire model as this would be expensive in terms of time and computer memory. This way, only a categorical checking is employed for the specific requirements using a relevant algorithm. By this approach, the problems of state explosion which limits application of model checking are mitigated.

Chapter 8

Compliance Checking and Verification

8.1 Introduction

The chapter presents the application of the artifacts, i.e. the compliance verification algorithms to check the compliance of a business process with the required constraints.The formalisation and the design of the compliance verification algorithm followed a step wise approach based on use case 1 which was described in section 4.2 in chapter 4. To demonstrate artifact applicability, we still apply use case 1 but in a different way. Use case 2 is used to evaluate the artifacts in Chapter 9. For this purpose, understandability and space reasons, use case 1 is abstracted to represent internal process operations of the store, and verified using the overall compliance verification algorithm in section 8.4 Specifically, the order processing instance is considered. The rest of the chapter is structured as follows: Section 8.2 presents the revised use case, applicable requirements and constraints. Section 8.3 lists a set of compliance requirements, shows their translation into constraints through formal expressions using DL and LTL, and how they can be verified based on a verification scenario. Section 8.4 illustrates the application of the overall compliance verification algorithm to verify the entire business process. Section 8.5 presents a discussion based on the verification outcomes while Section 8.6 presents chapter summary.

8.2 The Abstracted Pick and Pack Use Case

The process starts with arrival of orders in the store's order catalogue. The orders are sorted, assigned and processed to completion. The order processing Eco system is composed of the orders, customers, staff and, policies and regulations, regulatory agencies among others. These play different roles;

- Orders are placed by customers and pick them when they are ready or wait for delivery.
- Staff process orders at the store e.g. Pickers, Packers, supervisors among others.
- Policies and Regulations guide operations of the business process.
- Regulatory agencies specify and monitor enforcement of policies and regulations.

The activities in the abstracted pick and pack business process are briefly described as follows;

- Select Order (*So*): the order is selected from the pending orders by a staff who will process it. This is the initial activity which signals the start of order processing instance.
- Pick items (*Pit*): The items are picked by the store staff. A store may have one or more store departments and staff may cross between departments or are restricted to one.
- Verify order (*Vo*): This is a quality check to ensure the order is fulfilled in terms of the right items and quantities.
- Pack order (*Po*): The order is packed and made ready for delivery or pickup by the customer.
- Hand over (*Ho*): The ready order is handed over to customer service unit
- Customer Pick up or Delivery (*Cpd*): if the order is not picked up the delivery staff will deliver the item within the specified duration.

Based on the process activity brief description above, Consequently the model in Figure 8.1 is realised.

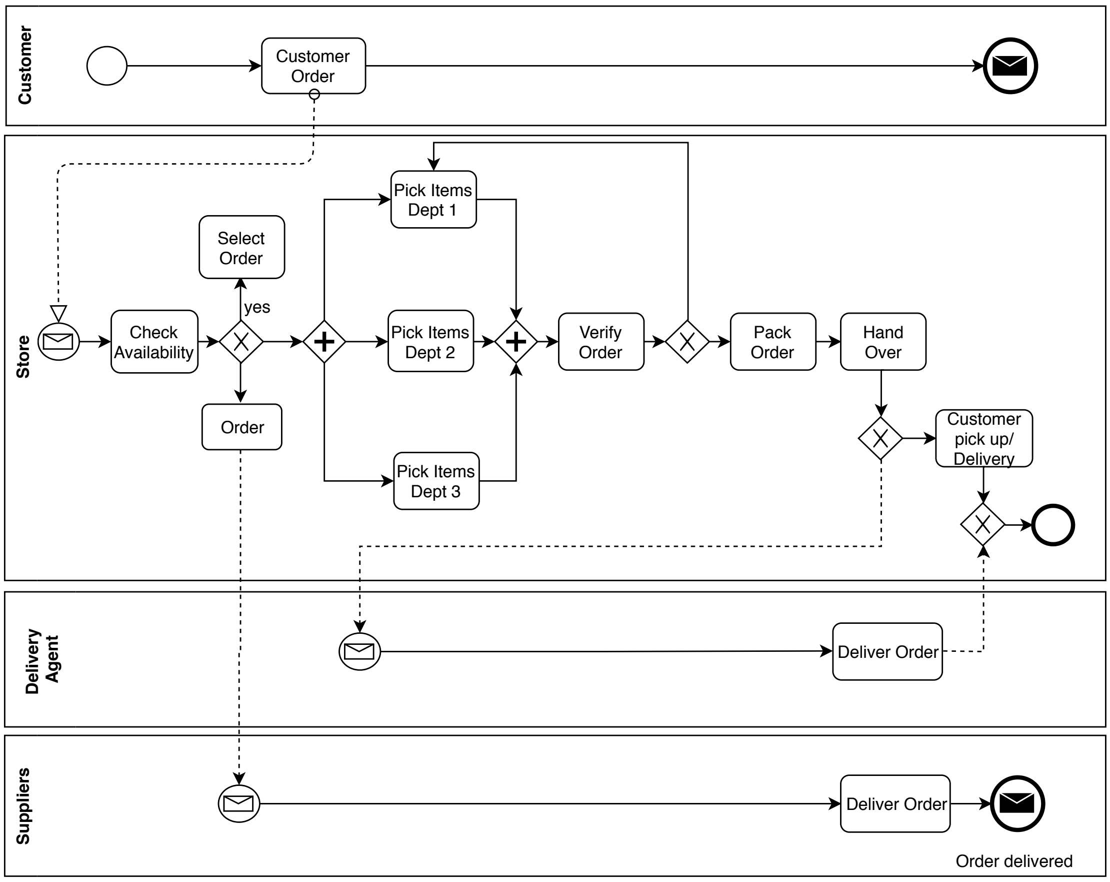


Figure 8.1: Abstracted pick and pack business process model

8.2.1 The Internal Requirements of the Business Process

As described, the business process must conform to a set of policies specific for a store. Some of the relevant policies include;

Control flow and temporal policies to guide process executions are as follows;

1. Each order must start with the select order activity and end with customer pick up or delivery. The total order processing time is 3 hours.
2. During order processing, big orders are picked by more than one staff. This activity duration should not exceed one hour.
3. Every order must be verified before it is packed. Verification of each order depending on the size within 20 minutes.
4. Packed orders are ready for handover to customer service section.

5. Orders are picked by customers or delivered to customer premises. Delivery takes one hour whereas the customers must pick their orders within a day otherwise they are put in the storage.

In addition, resource based policies to guide allocation resources are as follows;

- Pickers are allocated to pick items and cannot execute verify orders.
- Packers are allocated to pack order task. However, they also execute verify order task.
- Pickers can be delegated to participate in order hand over to customers if they are free or when there are high volumes.
- Supervisors oversee other employees and can execute any task.
- Supervisors can execute delegate tasks. E.g. supervisors can delegate pickers to pack items

The specified tasks are executed if access to necessary data is provided. To this effect, policies to guide access control to data are specified as follows;

- Supervisors have full access to data and can grant data access to staff based on organisational roles and tasks they execute in the business process.
- Basic data must be accessible and available for staff to execute tasks that do not need much restriction and control. For example, order list data should be accessible and available to pickers, verifiers and packers.
- Access control and authorisation must be observed for data privacy. For example, customer personal data, financial data among others
- Customer data is considered as private data to which the principle of privacy must be observed.
- Security of the data and system is important and worth observation. To this effect, users and staff must be authenticated to use the system.

The internal policies are superseded by the external regulations. The super store being cross regional, several external regulations apply. Such as;

- The European union general data protection act (GDPR) which emphasizes data privacy and security.

- The Sarbanes Oxley Act (SOX) which emphasizes the separation of duty and binding of duty.
- The UK consumer protection act which emphasizes consumer protection rights like right to quality products and services, right to return goods, right to be refunded.
- The Health Insurance Portability and Accountability Act (HIPAA) or the NHS equivalent which defines basic security and privacy practices for health care and pharmaceutical dispensaries. The act applies to the stores since many of them operate pharmacies.
- Trade laws limiting sale of restricted products to specific groups of customers like those in under age category. For example, sale of alcoholic products. Also, sale of health products that require drug prescriptions.
- Service level agreements for acceptable business transactions and customer relations.

Both internal policies and external regulations must be complied with by the business process. Because of the collaboration, contractual obligations are composed and agreed upon by the parties as guiding principles for business operations. A collection of requirements from applicable policies, rules, laws, standards and regulations forms a set of all compliance requirements that the business process must conform with.This document is updated as change in policies and regulations occur.

As earlier indicated, policies and regulations are stated in natural language and thus bound to suffer the challenges of natural language such as ambiguities and inconsistency. The extracted requirements form the compliance constraints that are verified with the business process model. The verification is only possible with formalised constraints. From this point, the artifacts put forward by this thesis are applied. In the next sections, the application of constraint expression mechanism is illustrated.

8.2.2 Constraint Elicitation and Expressions

In consideration of the above, a list of requirements and constraints are for the pick and pack process is presented in tables in section A.2 of appendix A. The next step is to formalise the listed constraints through formal specifications in section 8.3 based on DL.

8.3 Requirements Expressions

8.3.1 DL Based Specifications

Following the constraint expression mechanism which was described in chapter 5, this section illustrates requirements representations using DL. The symbols used include;

- $\sqcap$ Conjunction of constraints
- $\sqcup$ Disjunction of constraints
- $\rightarrow$ Assignment of an activity to a constraint
- : Assignment of subsequent constraints after the initial (control flow) constraint
- [,] Brackets holding constraint attributes

Constraint Representations sing Unary Expressions

The unary expressions represent individual category-based constraints;

(a) *Exemplified control flow and temporal constraint expressions* Requirement 1 specifying that the select order activity Starts every order processing instance, executed within 10 minutes, assigned to Pickers but can be delegated and data access is limited access to order catalogue. This requirement can be expressed as follows:

$So \rightarrow (Exist) \sqcap Duration : (10mins)$

$Pit \rightarrow [So]Precede \sqcap$ BoundedExit $^{(n-1)}$ $\sqcap Duration : (20 - 50mins)$

$Vo \rightarrow [Pit]Precede \sqcap BoundedExit[n] \rightarrow Duration : (\leq 20mins)$

$Po \rightarrow [Vo]Response \sqcap Precede \sqcap Valid : (10mins)$

$Ho \rightarrow [Po]Precede \sqcap Delay : (20mins)$

$Cpd \rightarrow [Ho]Precede \sqcap BoundedExit[n] \sqcap (Duration : [1 - 2hrs] \sqcap Repetition : [10mins])$

(b) Exemplified Resource constraint expressions

$So \rightarrow (Supervisor) \sqcap Delegate : (Supervisor \rightarrow Pickers)$

$Pit \rightarrow (Pickers, Supervisors) \sqcap Delegate : (Supervisor \rightarrow Packers)$

$Vo \rightarrow SoD : (Supervisors, \neg Pickers) \sqcap Delegate : (Supervisor)$

$Po \rightarrow BoD : (Supervisors, Packers)$

$Ho \rightarrow BoD(Supervisors, Deliverystaff)$

$Cpd \rightarrow BoD(Supervisors, Deliverystaff)$

(c) Exemplified Data constraint expressions

$So \rightarrow Prv \sqcap Authentication : (Ordercatalogue)$

$Pit \rightarrow AA : (Itemorderlists) \sqcap Prv : (Departmentitemlists)$

$Vo \rightarrow Prv \sqcap Authentication : (Itemorderlists)$

$Po \rightarrow Authentication(Ordercatalogue)$

$Ho : (Ordercatalogue)$

$Cpd \rightarrow Visible \sqcap AA : (OrderCatalogue) \sqcap Privacy : (Customeraddress)$

Constraint Representations Using Binary Expressions Binary expressions are composite representations involving combinations between sets of constraints. The requirements in Table ?? involve combinations of constraints that guide execution behaviour. This subsection illustrates expression of requirements involving binary constraints per activity.

(a) Select order execution constraints expression

$So \rightarrow (Exist \sqcap \neg Precede) \sqcap Duration : [< 10mins] \sqcap BoD : [Picker] \sqcap Itemorderlist : [Auth] \sqcap [Prv]$

Requirement 1 specifying that the select order activity Starts every order processing instance, executed within 10 minutes, assigned to Pickers as BoD but can be delegated and data access is limited access to order catalogue by access control and authorisation.

(b) Expressions of Pick items execution requirements

$Pit \rightarrow (\neg Exist[So] \sqcap BoundedExist[n_{(n-1)}]) \sqcap Duration : [20 - 50Mins] \sqcap (BoD : [Picker] \sqcap [Delegate : (Supervisors, Picker, Packer)) \sqcap itemorderlist : [AA] \sqcap [Prv]$

The expression specifies that pick items activity is preceded by select order and can be repeated several times until all items on the order list are picked. The scheduled duration is between 20 and 50 minutes, with a BoD resource constraint for the picker, and access to item order list data granted by access and availability, and by access control and authorisation.

(c) Expressions of Verify order execution requirements

$Vo \rightarrow (Precede[Pit] \sqcap BoundedExist[n_{(n-1)}]) \sqcap Duration : [< 20Mins] \sqcap (SoD : [\neg Pickers] \sqcap Delegate : (Supervisors, Packer)) \sqcap itemorderlist : ([AA] \sqcap [Auth])$

The expression specifies that verify order activity is preceded by Pick items and its conditions must be satisfied before the process continues to the next level which implies that it is repeated several times. The scheduled duration is less than 20 minutes, with SoD resource constraint for the pickers and supervisor who can delegate to pickers. Access to item order list data is granted by authentication, and by access control and authorisation.

(d) Pack Order execution constraints expression

$Po \rightarrow (Precede[Vo] \sqcap Response) \sqcap Valid[= 30Mins] \sqcap (BoD : [Packers] \sqcap Delegate[Supervisors, Picker] \sqcap itemorderlist : ([AA] \sqcap [Auth])$

The expression specifies that pack order activity is preceded by verify order and occurs as a response to verify order. Its execution is valid for 30 minutes. The assigned resource constraint is BoD for the packers and supervisor who can delegate to pickers. Access to item order list data is granted by accessibility and availability, and access control and authorisation.

(e) Handover Order execution constraints expression $Ho \rightarrow (Exist \sqcap Precede[Po]) \sqcap Delay[20Mins] \sqcap Role : [Supervisors, DeliveryStaff] \sqcap itemorderlist : [AA] \sqcap [Prv]$

The expression specifies that handover order activity is preceded by Pack order. Its execution is delayed for 30 minutes to allow batch processing of handover. The assigned resources are supervisors and delivery staff. Access to item order list data is granted by accessibility and availability, and by authentication.

(f) Customer pick-up or Delivery execution constraints expression $Cpd \rightarrow (Exist \sqcap Precede[Po]) \sqcap (Duration : [1-2HoursMins] \sqcap Repetition[10mins]) \sqcap [Supervisors, DeliveryS$ $(itemorderlist : [AA], customeraddresses : \sqcap[Prv])$ The expression specifies that order delivery or customer pick-up activity is preceded by handover order, executed for a duration of 1-2 hours and it is repeated every 10 minutes in case the order is rejected. The assigned resources are supervisors and delivery staff with access to order list data granted by accessibility and availability, while customer address data is granted by satisfying privacy data constraints.

8.3.2 Exemplified Formal Constraints

To enhance the reasoning capacity, DL was extended with integration of basic constructs of LTL i.e. operators and quantifiers to obtain more formal constraint expressions. The model logic created facilitates compliance verification and checking of business process and constraints. The section below presents the exemplified formal expressions.

(a) Select order execution constraint expression

$G(So[init] \wedge [< \ 10mins] \wedge [Picker, Supervisor \ : \ BoD] \wedge [Itemorderlist \ : (AA, Auth]$

The expression specifies *So* as an initial activity whose duration is less than 10 minutes. It is assigned to pickers and supervisor as resources constrained by BoD which implies that the picker can participate in another activity. Access to item order data is controlled by access and availability as well as authentication.

(b) Pick Items execution constraint expression

$G(Pit^{n}\xrightarrow{(n-1)} \wedge[20{-}50Mins]\wedge[Picker : BoD, (Supervisors, Packer : Delegate)]\wedge [itemorderlist : (AA, Prv)]$

The expression specifies *Pit* as an activity that can be repeated for *n* times, for a duration between 20-50 minutes. It is assigned to pickers and supervisor as resources constrained by BoD which implies that the picker can participate in another activity. The supervisor can delegate task to packers. Access to item order list data is controlled by access and availability as well as authentication.

(c) Verify order execution requirements

$G(Vo^{n}\xrightarrow{(n-1)} \wedge[< 20Mins]\wedge[(Verifiers[SoD])(Supervisors, Packers : [Delegate])\wedge itemorderlist : (AA, Auth)])$

The expression specifies *Vo* as an activity that can be repeated for *n* times until it passes, for a duration between of less than 20 minutes. It is assigned to packers as resource constrained by SoD. The supervisor can delegate task to packers. Access to item order list data is controlled by access and availability as well as authentication.

(d) Pack Order execution constraint expression

$G(Po \rightarrow \wedge[30Mins] \wedge [Packers : BoDSupervisors, Picker : Delegate] \wedge [itemorderlist] : (AA, Auth))$

The expression specifies *Po* as an activity to be executed for a duration of 30 minutes or less by packers and supervisor as resources constrained by BoD which implies that the packers execute Po in relation to another activity. The supervisor can delegate the activity to pickers. Access to item order list data is controlled by access and availability as well as authentication.

(e) Handover Order execution constraint expression

$G(Ho \rightarrow [20Mins] \wedge$[(Supervisors), Pickers: Delegate]$\wedge[itemorderlist$:(AA,Auth)])

The expression specifies *Ho* as an activity scheduled for a duration of 20 minutes. It is assigned to supervisors who can delegate to pickers. Access to item order list data is controlled by access and availability as well as authentication.

(f) Customer pick-up or Delivery execution constraint expression

$G(Cpd \rightarrow \wedge[1 - 2HoursMins, 10Mins] \wedge [Supervisors, DeliveryStaff] \wedge [itemorderlist : AA, customeraddresses : prv])$

The expression specifies *Cpd* as an activity scheduled for a duration between 1- 2 hours. It is assigned to supervisors and delivery staff. Access to item order list data is controlled by access and availability while customer addresses data is controlled by privacy constraint as well as authentication.

(g) **if** Duration $>=$24 hours **then** Action " Take package to store"

When the orders are not picked for the day, they are taken to the store for storage. The expressions in this section demonstrate the converted formal expressions making use of binary relations among the constraints to specify behaviour of the process.

To illustrate the reasoning, a set of verification requirements are specified as follows;

8.3.3 Verification Scenario - Requirements

In this scenario, the following verification requirements are listed, their specification and formal expressions;

1. Every order processing instance starts with select order and ends with delivery or customer pick up.

   $G((So), F(Cpd))$

   For purpose of checking termination of instances, each terminating case starts with select order and ends with order delivery or pickup.

2. Every order processing instance must be verified. Verify order must exist in every instance.

   $G(\forall \sigma \in Pi \exists Vo)$

   For every case of order processing instance must always be verified

3. Supervisors have rights to every task and can delegate tasks to other users.

   $G(\forall Activities, Supervisor \rightarrow (Prv.[Read]) \wedge F(Delegate))$

   For each activity, always the supervisor has access control and authorisation, and can eventually delegate permissions.

4. A set of activities are BoD and SoD respectively

   $G((Pickers, Supervisors).BoD \rightarrow (So, Pit)$

   Activities select order and Pick item are always executed by resource actors pickers and supervisors constrained as BoD. The roles meet resource actors selection conditions for the execution of *So* and *Pit*.

   G((Verifiers,Supervisors)$\wedge(\neg Pickers).SoD \rightarrow (Vo)$

Activity verify order is always executed by verifiers or supervisors as designated role actors that meet resource selection conditions for its execution. Pickers are excluded from roles that can execute verify order.

5. Verify Order must wait until Pick order is completed. Pick order is repeated until all items are picked.

$G((Vo)W(\sum_{(\ n-1)}^{n} Pit^{n}) \rightarrow n = k)$

Verify order must wait until pick items executes for a specified number of times i.e. until all items are picked where $k =$ number of items.

6. Where stock of items is not available for an order, suspend order and contact customer

   $G(\sum_{(\ n+1_{n})}^{n} Pit, F(Suspend \wedge Contactcustomer))$

   If the items picked do not sum up to the items ordered (if no more items are available), the order is suspended and customer is contacted.

7. Unavailable items can be substituted upon permission from the customer

   G(Pit $\rightarrow [Item{-}unavailable], (Contactcustomer{\wedge}Replace){\vee}F(alternativeitemsatdelivery))$

   Where items on the order are not available, the customer is contacted to replace the items or alternative items are carried and offered during the order delivery.

8. The total order processing time is approximately 3 hours. The total duration for processing each case of the order is given by;

   Total process duration=$(\frac{(\sum(a_1 \ldots a_n)+delays)}{(\sum t_n)})$

   Duration=

   $$\sum_{t}(So, Pit, Vo, Po, Ho, Cpd)$$

   Using the formal specified verification requirements,the next section shows how to check for their fulfilment and compliance through application of the verification algorithms.

8.4 Application of Compliance Verification Algorithms

To verify the business process's compliance with above constraints, the overall compliance verification algorithm 12 is applied.

8.5 Discussion

Algorithm 12 is applied to check compliance to verify requirements enlisted in section. The specific properties verified in this case include the following; Termination property: this property is used to check the possibility that a model has start and end points, i.e. a model can start and end. To check this property, the algorithm 12 checks for existence of initial and end activity events for each complete case in a process instance. Absence of initial and end events indicates lack of termination which is also a source of deadlocks i.e. tasks that start and never complete. It also violates the constraints for initial and end activities specified in requirement 1.

Deadlocks: checking for this deadlocks in models ensures that no activities remain stuck, incomplete or unexecuted due to lack of resources, resource overutilisation or unintended lock out or denial to data access. For example, due to SoD restrictions, it situations may arise where no resource is available to execute a task. The algorithm checks to detect deadlocks likely to be caused by resource allocation. This is enforced by checking constraints related to resource allocation to process activities such that deviant behaviour leading to violations can be detected early in time. From the use case, at least the supervisor role is assigned to each task as a continuity strategy. The algorithm further checks for the existence of roles that can free over allocated resources or execute tasks that may exist without assigned resources or whose resources may be busy. From the use case, the supervisor role is assigned for each task as specified in requirement 3, thus the algorithm checks for its existence. The non-existence of supervisor role assignment over tasks is considered a violation.

Livelocks: checking for livelock in the model ensures that no instances are trapped in infinite loops. For example sources of livelocks in the use case are; orders that remain pending because of non-availability of stock items, orders that do not pass verification and executions that remain pending due to denied data access. Specification 7 allows item substitution where an ordered item is not available. This helps to prevent order suspension which is a likely source of livelocks. The algorithm in this case will verify for existence and permission to execute the substitute item activity in the model. Absence or lack of necessary resource assignments to execute this activity amounts to a violation.

Temporal conflicts checking: the verification of temporal constraints checks for conflicts related to temporal assignments where resources (roles) may be assigned to different tasks whose execution occurs at the same time, or activities that start and end at the same time yet assigned to same resource. This would imply that only one task may be attended to due to conflicts in execution time causing a delay in the entire process duration. The algorithm checks for conformance to temporal requirements and detecting likely deviations based on the total process duration. Where the duration is beyond the total activity scheduled times, it implies a delay. The algorithm will proceed to check and identify the activities likely to cause delays and thus violating the temporal constraints. Requirement 8 specifies total order process instance duration to be 3 hours. The algorithm sums up the specific activity durations and delays to determine the compliancy to the required process cycle time. If the execution time exceeds the scheduled time, then a temporal violation is reported.

Permission lock Property: the property relates to checking of conflicts relating to access control and authorisations where permissions may be granted and denied at the same time, or permit and authorise the same role for the same activity at the same time. This leads to permission locks which the algorithm assists to identify by assessing the data constraint assignments concerning access control and authorisation, security and privacy. From the case, access to data requires access and availability for the specific assigned roles except where customer data which is considered private as requirement 6 and 7 specify. Access to customer addresses is controlled by privacy constraint. The algorithm checks for compliance to this constraint. To facilitate further evaluation of the artifacts outcomes, a practical implementation of a prototype is necessary.

8.6 Chapter Summary

Compliance verification involves checking a model's conformance to specified constraints. This chapter was dedicated to demonstrating application of the proposed compliancy approach in chapter 7. Based on the abstracted industry based use case, this chapter showed how to;

- Extract compliance requirements into a single composed document.
- Use the proposed DL semantics, the requirements were formalised into constraints and further expressed into a formal structure to support reasoning as a necessary condition to achieve verification through model checking.
- Apply the proposed algorithms to verify for compliance of the business process with constraints through violation detection.

Generally, the chapter is dedicated to illustration of application and efficacy of the artifacts which are the outcome of the research. However, the algorithms are generic to accommodate wide applications without limit to a specific area of application. This is further illustrated with evaluation based on use case 2 in chapter 9. In the following chapter, an evaluation is conducted to rigorously assess the expressivity, efficacy and efficiency of the presented artifacts.

Algorithm 12 Overall Compliance Verification - Pick and Pack Business Process

```
1: $InPut$:
       BP, Constraints
2: for all $Pi$:C.(e.ac) = init, end do
       init = So, end = Cpd
3:     if init $\in seen, finished \neq$ So then
       "Violation of Start activity constraint."
4:         if end $\in seen, finished \neq Cpd$ then
       "Violation of End activity constraint."
5:         end if
6:     end if
7: end for
8: Verify constraints BoD, SoD and Delegate
9: for all actors $r(Pickers, Packers, Verifiers, Supervisors) \in R$ do
       $(Pickers)$.BoD = $((So, Pit), Pickers)$
       $(Verifiers, Packers)$.SoD = $(Vo, Verifiers) \wedge (Vo', Packers), \neg Pickers)$
       $(Supervisor, Pickers)$.Delegate = $(Vo, (Verifiers \wedge Pickers))$
10:    if $(Packers).BoD \in seen, finished \neq ((So, Pit), Pickers)$ then
       "Violation of BoD constraint for Pickers over" $e.ac = (So, Pit)$
11:        if $(Verifiers, Packers).SoD \in seen, finished \neq (Vo, (Verifiers \wedge Packers))$ then
       "Violation of SoD constraint for Verifiers and Packers over Vo"
12:            if $(Supervisor, Pickers)$.Delegate $\in seen, finished \neq$(Vo,(Verifiers $\wedge Pickers)$) then
       "Violation of Delegate constraint for Supervisor and Pickers over Vo"
13:            end if
14:        end if
15:    end if
16: end for
17: Verify for existence of verify order for each $Pi$
18: for each $Pi$, C= Exist.Vo $\in \sigma$ do
       $\forall \sigma \in Pi \rightarrow \exists$ (Vo, Supervisor) =0
```

```
19:       if $\forall Pi \nexists$ Exist.Vo then
            ”Constraint Violation for Exist.Vo”
20:           Verify for data constraint compliance $Pi$
21:           if Order = ”Suspended” then
            $e.ac =$ Contact Customer, supervisor,[Read.customer data = authorise]
22:           else
            ”Violation of data constraint, denied access to customer contact data”
23:           end if
24:       end if
25:   end for
26:   Verify compliance with Temporal constraints $Pi$
27:   if Total $Pi$ Duration $\neq<$ $\sum$ e.ac $\in$Pi then
        ”Violation: Instance delay detected”
28: end if
29: Feedback
        No Violation for start and end activity constraints for the provided business process.
        No Violation of BoD constraint for the provided business process.
        No Violation of SoD constraint for the provided business process.
        No Violation of Delegate constraint for the provided business process.
        No Violation of Existence of Verify order constraint for all instances in the process.
        No Violation of assignment of supervisor actor for all instances in the business process.
        No Violation of temporal constraint for the provided business process.
```

Chapter 9

Evaluation and Discussion

9.1 Introduction

Previous Chapters showed application of our approaches using an example to show how the our compliance verification approach works. In this chapter, we use a collaborative business process to evaluate the design method and verification algorithms. A known evaluation model, the method evaluation model (MEM) is also used to assess the efficacy of the artifacts. The rest of the chapter is presented as follows. In section 9.2, the MEM model is introduced and and used to evaluate the efficacy of our compliance verification approach. While section 10.3 the efficiency of the algorithms is evaluated using a second use case. The performance of the algorithms is evaluated in section 9.3, and last section 9.6 summarises and concludes the chapter.

9.2 The Adopted Evaluation Model

Traditionally, different methods are applied to evaluate Information system designs. For example, the method evaluation model (MEM) which employs user perceptions and performance and intentions, and behaviour of users to evaluate models methodologically [108]. Figure 9.1 illustrates the MEM. Following the variables in MEM, a model is assessed in terms of the following parameters;

- Efficiency; the expected performance of the system
- Efficacy: expected benefit from using the system
- User perception of efficiency and efficacy of the model

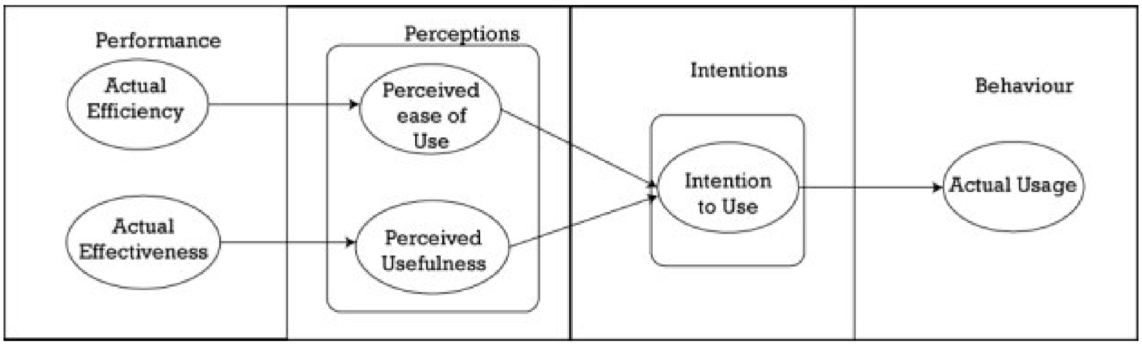


Figure 9.1: MEM model [108]

The efficacy parameter has been applied to evaluate the artifacts put forward in the thesis to illustrate their expected benefits. The constraint expression mechanism i.e. the DL language is a less complex compliance requirement definition and constraint specification mechanism, with semantics and syntax is close to natural language. With its application, it is possible to express constraints naturally yet formal enough. The expected benefit is the simplified process of compliance definition and specification, while facilitating their comprehension and comprehensibility for non-experts.

More still, the compliance verification using the designed algorithms is simplified because of the iterative approach in which they are designed. For each constraint category, specific algorithms are designed as well as the overall algorithm. This makes the verification process highly flexible in a sense that verification can be targeted to a specific constraint without having to check the entire process, or otherwise. Moreover, the intelligible feedback returned pointing to the source of the violation is easily comprehensible for the end users. Besides the efficacy, the algorithms are as well evaluated in terms of performance capacity. The following section introduces algorithm performance evaluation.

9.3 Performance Evaluation of the Algorithms

In general, performance evaluation of algorithms involves assessing their capacity in terms of different parameters like scalability and efficiency. The most common algorithm performance measurement indicator is the computation complexity associated with the algorithm. Algorithm complexity is a function $f(n)$ for measuring time and space used by an algorithm in terms of input size n. It is a mechanism to classify an algorithm's efficiency compared to alternatives.

An algorithm's complexity is computed in terms of time and space as resources required by the algorithm to run. The amount of resources required by the algorithm depends on how much of the Input resources are available to produce the output. To derive the required resources therefore, a general function is applicable;

$$n \to f(n)$$

where

n = the Input size

f(n) = the worst-case complexity or average case complexity of the algorithm.
Worst-case complexity refers to the maximum amount of resources whereas average case complexity is the average amount of required resources for Input of size *n*.

9.3.1 Time Complexity

Algorithm time complexity $T(n)$ refers to the amount of time required by an algorithm to run computed by counting elementary operations. It is expressed in terms of steps or operations through which the algorithm processes the Inputs to produce output. For each operation, an estimate of required time is defined and assumed to be constant for all steps. Since required time will vary with input, increasing input sizes are relevant to compute time, given by the function:

$$\mathcal{O}(n)$$

where $\mathcal{O}$ = the time and n is the input size.
If the computations are achievable in feasible time for deterministic Inputs, then the algorithm is said to be of polynomial time and tractable.
In relation to the presented algorithms, the number of steps taken by the constraint checking algorithm while verifying for compliance determines how much time is required. To achieve verification in polynomial time, compact inputs by means of constraint based checking is employed. For example, from the control flow constraint category, a specific constraint like dependency is checked each time. Besides,for each time verification is conducted, only relevant constraints are checked as opposed to complete model verification, unless when required otherwise. This approach to constraint verification is considered efficient and feasible in time and thus tractable for the deterministic finite states.
The time computation depends on the input sizes to the elementary operation of the algorithm.

The elementary operation is that operation which the algorithm performs. In our case the algorithms detect violations by comparing modelled behaviour and observed behaviour based on generated traces. Therefore the elementary operation involves behaviour comparison which dominates other operations. The comparison time is assumed constant regardless of the input size of the constraints being checked. The time complexity of each algorithm is therefore given by;

$$T(n) = n - 1 \tag{9.1}$$

Therefore, by assuming that each step in violation detection has the size $O(1)$, the overall complexity is given by

$$\mathcal{O}(n^n) \tag{9.2}$$

Worst-case Time Complexity

To tell whether an algorithm detects a violation depends on the number of steps in the elementary comparison operation. If a violation is not detected, then n comparisons are made to check all states in a trace. Otherwise one comparison is made upon violation detection.
To compute the worst case time complexity,

- Let $T_{1(n)}, T_{2(n)}, \dots$ be the checking times for all possible Inputs of size n.
- The worst case time complexity $W(n)$ is then given as $W(n) = maxT_{1(n)}, T_{2(n)}$.
  The worst-case time complexity for the algorithm therefore i;

$$W(n) = n \tag{9.3}$$

The Average-case Time Complexity

The computation of average-case time complexity involves both possible Inputs of size n and the probabilities of the Inputs. Therefore;

- $T_{1(n)}, T_{2(n)}, \dots$ are comparison steps for all possible Inputs of size n,
- $P_{1(n)}, P_{2(n)}, \dots$ are Input probabilities Average-case time complexity is therefore given by

$$P_{1(n)} + P_{2(n)} + \dots \tag{9.4}$$

9.3.2 Space Complexity

Space Complexity refers to computation requirements of the algorithm in terms of amount of memory space required to compute the Input to realise the output.

9.4 Artifact Evaluation Based on Use Case 2

Use case 2 is adopted from [166] and used to evaluate the algorithms put forward in the thesis. To facilitate constraints checking based on the car insurance use case, a car insurance collaborative trading business process adapted is to an abstracted model shown in Figure 9.2. The model illustrates a collaborative business process executed by different stakeholders interacting at different levels. The process starts when a policy holder makes a car insurance claim. The claim is registered by the Euro Assist (insurance broker) company which assigns a garage and contacts the insurance company. AGFIL, the insurance company when contacted forwards the claim to Lee C.S which appoints an Assessor to assess the car damage. After assessment the car garage is contacted which send repair cost estimates, repairs the car and forwards the invoice through Lee C.S to the AGFIL which pays all invoices.

9.4.1 Use Case Adaptation and Application

From their original work [166, 168], the use case was used as a case to support contract monitoring by detecting and guarding against violation by partners. Basically, the work supported contract fulfilment by monitoring actions of contract partners in an e-contract. However, their work does not detect any flaws within the business process itself. In this thesis, we use the case to evaluate the compliance verification algorithms by checking and detecting violations to compliance constraints in the case as defined in Chapter 7. To facilitate the checking, sample data in Table 9.1 is used as baseline data showing normal activities and events according to the requirements.

Based on chained execution as a collaborative business process interaction mechanism where each partner concentrates on a sub process, the car insurance trading business process is composed of sub processes (see model 9.2) in which each partner is responsible for a specific component. Table 9.1 shows process activities and events according to a control flow arrangement and party actors as resources. The business process must conform to a set of constraints derived from policies and regulations governing general business and car insurance trade. For illustration

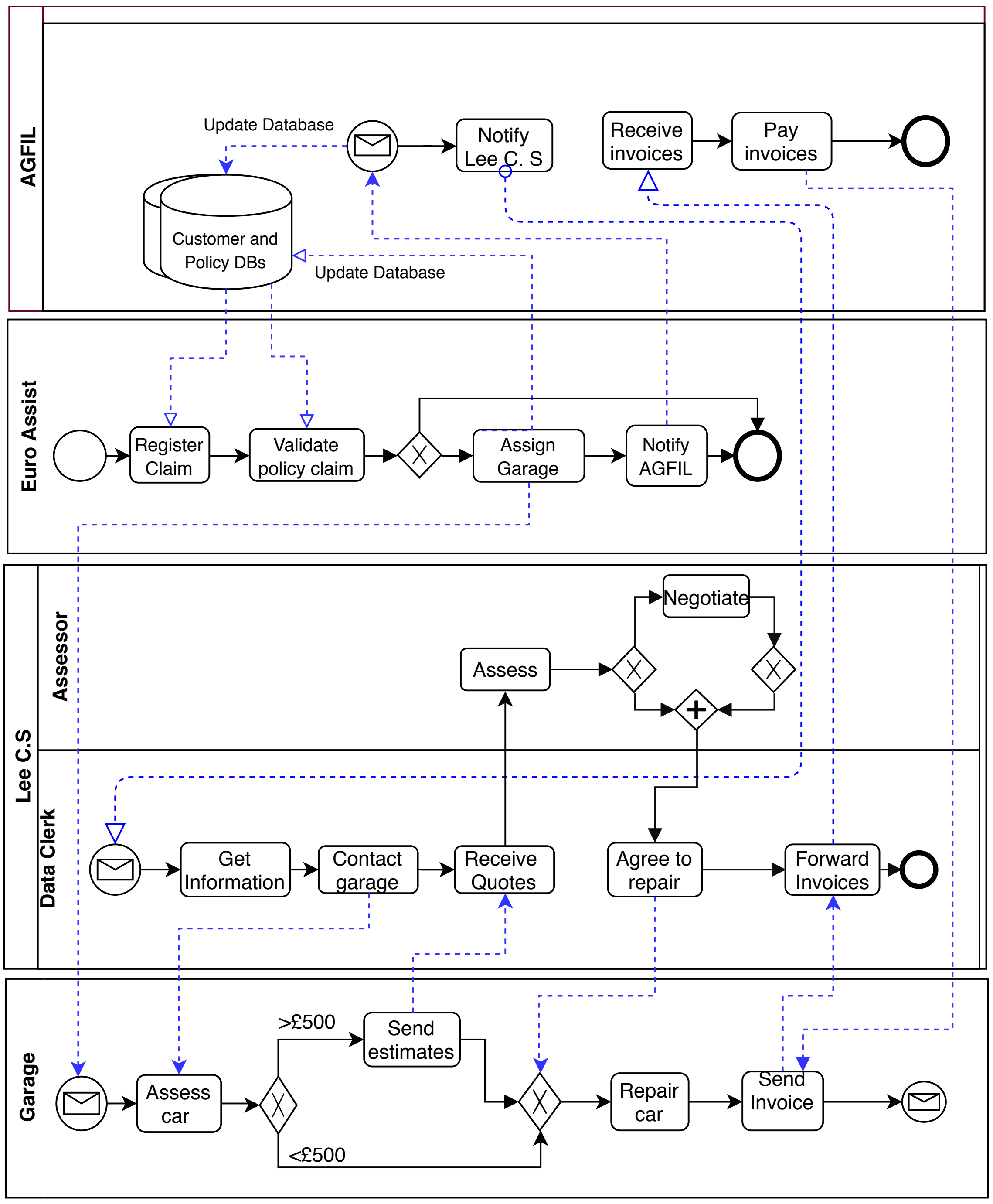


Figure 9.2: Abstracted Car insurance collaborative trading business process

purposes, we limit the constraints to the following:

9.4.2 Car Trading Business Process Constraint Requirements

(a) Each complete process instance starts with register claim activity and ends with payment for the raised claim (pay invoices).

(b) There is a high level dependency and response between the activities. Completion of an activity at the local level leads to start of corresponding activity at another level. Any break in the Precedence or response between activities is a violation. For example, Pay invoice is preceded by receive invoices. Invoices are paid in a batch for all received for a period of one month. Each calendar month is 30 days.

(c) Activities assign garage and contact AGFIL are executed as BoD

(d) The user who receives invoices in AGFIL sub process should be different from the one that pays the invoices because they should be subject to auditing to prevent fraud.

(e) It is a regulatory requirement that a claim must be processed within 10 working days.

(f) For security and privacy, customer data must be protected from misuse.

Table 9.1: Abstracted Car Trading Process Activities

Activity	**Activity Id**	**Dependency**	**Actors**	**Resource Constraints**	**Data Constraints**
Register claim	a.1	-	Data clerk - Euro Assist	BoD with a.2	Privacy
Validate Claim and Policy	a.2	*a.1*	Claims Manager - Euro Assist	SoD with a.3	Authentication
Assign Garage	a.3	*a.2*	Data clerk - Euro Assist	SoD with a.2	Privacy
Contact AGFIL	a.4	*a.2*	Data clerk - Euro Assist	-	Authentication
Notify Lee C.S	a.5	*a.4*	Policy officer - AGFIL	-	Authentication
Send Estimates	a.6	*a.2*	Garage	-	Authentication
Assess Quotes	a.7	*a.6*	Assessor Lee C.S	-	Privacy
Negotiate	a.8	*a.7*	Assessor - Lee C.S	-	Authentication
Agree to repair	a.9	*a.8*	Garage	-	Access and availability
Repair car	a.10	*a.9*	Garage	-	-
Send Invoice	a.11	*a.10*	Garage	-	Access and availability
Forward and reconcile	a.12	*a.11*	Lee C.S - Auditor	-	-
Receive Invoices	a.13	*a.12*	Accountant - AGFIL	SoD with a.14	Authentication
Pay Claims	a.14	*a.13*	Chief Accountant - AGFIL	SoD with a.13	Authorisation

9.4.3 Process Instances

From the abstracted car trading business process, three key process instances are realised; 1. Insurance claim processing when the repair estimates are above £500 where negotiations have to be involved to an agreed repair cost and its variants, i.e. 2. when the repair cost is above £500 requiring no negotiation and, 3. when repair estimates are below £500. Table 9.2 and its sub tables represent the process instances. In process instance 1 (P_1) sub table (a), the repair estimates are above £500 requiring execution of negotiation until a repair cost is agree. Whereas in the second process instance (P_2) in sub table (b) repair cost estimates are above £500. However there is no negotiation involved. In the last process instance (c)P3, repair estimates are below £500. The policy to this effect requires automatic triggering of the repair task without need for negotiation. Activities *Assess quotes, Negotiate and Agree to Repair* are thus skipped. For the three instances, only one instance is permissible for each case.

Table 9.2: Variant Process Instances

(a) P1

Activity	**Events**
Register Claim	e1
Validate Claim	e2
Assign Garage	e3
Notify Garage	e4
Notify Lee C.S	e5
Get information	e6
Contact Garage	e7
Assess Car damage	e8
Send Estimates >£500	e9
Receive Estimates	e10
Assess Estimates	e11
Negotiate	e12
Agree to repair	e13
Repair car	e14
Send Invoices	e15
Forward Invoice	e16
Receive Invoice	e17
Pay Invoices	e18

(b) P2

Activity	**Events**
Register Claim	e19
Validate Claim	e20
Assign Garage	e21
Notify Garage	e22
Notify Lee C.S	e23
Get information	e24
Contact Garage	e25
Assess Car damage	e26
Send Estimates >£500	e27
Repair car	e28
Send Invoices	e29
Forward Invoice	e30
Receive Invoice	e31
Pay Invoices	e32

(c) P3

Activity	**Events**
Register Claim	e33
Validate Claim	e34
Assign Garage	e35
Notify Garage	e36
Notify Lee C.S	e37
Get information	e38
Contact Garage	e39
Assess Car damage	e40
Send Estimates <£500	e41
Repair car	e42
Send Invoices	e43
Forward Invoice	e44
Receive Invoice	e45
Pay Invoices	e46

9.5 Evaluation of Verification Algorithms

Based on data in Tables 9.1 and 9.2, we evaluate the composed verification algorithms from chapter 7 using the business process described in use case 2 with constraints stated in section 9.4.2. The application and evaluation of the algorithms to verify compliance requirements of a different scenario (use case 2) is an attestation of wide application of our artifacts. Based on the evaluation outcomes, it shows that algorithms are applicable to different business scenarios. We can therefore conclude about their usefulness, applicability to various industrial cases and effectiveness in supporting compliance verification.

i. Algorithm 1 Evaluation - Basic Process Instance validity

Algorithm 1 checks the validity of the process model based on its start and end activity events for all process instances. According to constraint requirement (a) in section 9.4.2, each instance starts with register claim and ends with pay invoices. Algorithm 1 is evaluated as follows.

```
1: $Input$:
        a. Process Instances P1, P2 and P3
        b. Constraint - Basic Validity
2: for each $Pi \in BP$ do
        $e.ac = e.init, e.end$
3:    if $e.ac = (e_1, e_19, e_33[RegisterClaim]) \notin started, seen, finished$ then ”Violation:
      Instance validity violated for Register claim events in all $Pi$”
4:        if $e.ac = (e_{18}, e_{32}, e_{46}[payinvoices]) \notin started, seen, finished$ then
        ”Violation: Instance validity violated for pay invoice events in all $Pi$”
```

Otherwise, no violation of instance validity if the $registerclaim$ and $PayInvoice$ events $\in$ $started, seen, finished$ for all process instances

Performance evaluation of algorithm 1 in terms of time and space computation requirements reports less time and space required to compute the verification outcome.

ii. Evaluation of Algorithm 3 - Precedence Verification

Algorithm 3 verifies for compliance with precedence constraints (activities that must occur before others occur) for the business process as required by constraint (b) using traces in process instances given in Table 9.2. In this scenario, the constraint restricts that Pay Invoice must be preceded by Receive Invoices for a period of one month.

```
1: $Input$ :
        a. Process Instances P1, P2, P3
        b. Constraints (Precedence, validity)
2: for $Pi \in BP$ with Precedence (Prec) and validity (Val) constraints C do
        Prec =
        e.ac.($e_{17}$ [Receive Invoice]) $\Rightarrow$ e.ac.($e_{18}$ [Pay Invoice],[Val= >30 days])$\in$ (a)P1
        e.ac.($e_{31}$ [Receive Invoice]) $\Rightarrow$ e.ac.($e_{32}$ [Pay Invoice],[Val=>30 days]$\in$ (b)P2
        e.ac.($e_{45}$ [Receive Invoice]) $\Rightarrow$ e.ac.($e_{46}$ [Pay Invoice],[Val= >30 days])$\in$ (c)P3
3:    if $(Prec \notin seen, finished)$ then
        Violation: "Precedence constraint violated at different events of the process instances"
```

Otherwise, no violation of Precedence and validity constraints if $\forall Pi$ in the business process Receive Invoice $Precedes$ Pay Invoice

iii. Evaluation of Algorithm 4 - Response Verification

Response algorithm 4 verifies for activities whose occurrence is a response outcome of occurrence of the current activity. It detects violations if occurrence conditional activities do not induce the occurrence of the action activities. Response algorithm is evaluated based on response constraint requirement in (b) of sub section 9.4.2 as follows:

```
1: $Input$ :
        a. Process Instances P1, P2 and P3
        b. Constraints(Response)
2: for all $Pi \in BP$ with Response constraints C do
        Response =
        e.ac.($e_9$[Send Estimates >£500] $\wedge$e$_{11}$[Assess car Damage] $\wedge$e$_{12}$[Negotiate] $\wedge$e$_{13}$ [Agree
    to repair])$\mapsto$ $e_{14}$[Repair car]) $\in P1$
     e.ac.($e_{27}$[Send Estimates >£500]$\mapsto$($e_{28}$[Repair car]) $\in P2$
     e.ac.($e_{41}$[Send Estimates <£500]$\mapsto$ $e_{42}$[Repair car]) $\in P3$
3:    if (Repair car $\notin started, seen, finished$) then
        Violation: "Response constraint violated on repair car by the Garage"
4:    else if e.ac.Conditional activity $\not\mapsto$ e.ac.Action activity then
        Violation: "Response (Action) activity did not occur when the condition activity occurred
for all instances."
```

Otherwise, no Violation of response constraint if Conditional activity $\mapsto$ e.ac.Action activity for all or any of the instances in the business process.

iv. Evaluation of Algorithm 5 - SoD

The evaluation of SoD algorithm is based on role actors accountant and chief accountant constrained by SoD over Pay invoice activity as specified by constraint requirement (c) in 9.4.2.

```
1: $Input$:
       a. Process Instances P1, P2, P3
       b. Constraints (SoD)
2: for  all resource actors (Accountant, Chief Accountant) where constraint = SoD do
       e.ac.(Accountant, Chief Accountant).SoD $\rightarrow$ (Pay Invoice) $\equiv$ (Approve Payment, Chief
   Accountant), (Pay Invoice, Accountant) $\in Pi$
3:     if e.ac.(actors.Pay Invoice $\in$ seen, finished) $\neq$((Accountant, Chief Accountant).SoD) then
       " Violation: SoD violated for Accountant and Chief Accountant over Pay Invoice"
```

Otherwise, no violation of SoD constraint for the provided processes if actors (Accountant, Chief Accountant) for Pay Invoice events are $\in$ seen, finished. If actors are different to those assigned or do not exist, then a violation of the SoD constraint.

iv. Evaluation of Algorithm 6 - BoD

The evaluation is based on role actor data clerk at Euro Assist constrained by BoD for execution of register claim and contact garage as specified by the constraint requirement (d) and data in Table 9.1.

```
1: $Input$:
       a. Process Instances P1, P2, P3
2: for  each actor = Data Clerk where constraint = BoD do
       [Data Clerk].BoD $\rightarrow$e.ac.(Register Claim, Assign Garage) $\equiv$ (Register Claim,[Data Clerk])
   $\wedge(AssignGarage, [DataClerk]) \in Pi$
3:     if (Register Claim $\wedge$ Assign Garage) $\in$ seen, finished $\not\models$ [Data Clerk].BoD then
       Violation: "BoD violated for Data clerk over Register Claim and Assign Garage"
```

Otherwise, no violation of BoD constraint for the instances in the provided business process if Register Claim and Assign Garage $\in seen, finished \models [DataClerk].BoD$.

v. Evaluation of Algorithm 9 - Authentication

Verifying for constraint requirements specified in (f) require algorithm for checking authenticity. Algorithm 9 is evaluate as follows.

```
1: $Input$:
       a. Process Instances P1, P2, P3
       b. Constraints (Authentication [Auth])
2: for all data with constraint C = (Auth [True/False]) for actor = Data clerk do
       Assign $\equiv e_2$.(Validate Claim, [Data clerk]) $\rightarrow$ Auth = (Claims data:[True])
3:     if $(Assign \neq Auth \in seen, finished)$ then
       Violation: " Authenticated actors are denied access to restricted data."
4:         if $\exists actor(r_n) \notin$ Assign $\in seen, finished$ then
       Violation: "Access leakage, non-authenticated actor gained to restricted data."
```

Otherwise, no violation of Authenticity constraint for the instances in the provided business process if authenticated actors and activity gain access to restricted data as Auth ∈ seen, finished. Violations are detected where non-authenticated actors have access to data other than Data clerk or where authenticated actors are denied access to restricted data.

v. Evaluation of Algorithm 10 - Privacy

```
1: $Input$:
       a. Process Instances P1, P2, P3
       b. Constraints (Privacy)
2: for all data with constraint C = (Privacy [R/W]), actor= (Data Clerk, Assessor) do
       Assign $\equiv$ e.ac.($e_1$(Register claim) $\wedge e_3$(Assign garage),[Data Clerk]) $\rightarrow$ Privacy = Authorise:[R/W] $\in$ P1
       Assign $\equiv$ ($e_{11}$(Assess Estimates),[Assessor]$\rightarrow$ Privacy = Authorise:$[R/W/M]$ $\in$ P1
3:     if (Assign $\not\models$ Privacy $\in seen, finished$ ) then
       Violation: "Assigned actors and tasks denied access to private data "
4:         if $Assign \neq DataClerk, Assessor \in seen, finished$ then
       Violation: "Privacy leakage. Private data accessed by non authorised actors"
```

Otherwise, no violation of Privacy constraint for the instances of the provided business processes if Assign $\models$ Privacy $\in seen, finished$ i.e. no unauthorised resource actors gain access to private data or when no authorised actors or tasks are denied access to private data.

9.5.1 Time Performance Complexity of the Algorithms

Because the prototype is not yet fully implemented, the practical performance assessment of the time performance of the algorithms may not be realistic. In general terms, we can still approximate the time complexity requirements using the computation principles as described in section ??. The time requirements for each of the algorithms in the previous sections will depend on the time it takes each one of them to detect and report violations i.e. identify a constraint from the prescribed behaviour and compare it with the observed behaviour in the trace. The computation complexity will therefore consider the following attributes;

- Number of traces = n^T
- Number of constraints being checked = n^c
- Number of process instances = n^I

Therefore, the formulae below would enable time complexity computation for each algorithm by substituting with actual attribute values. ;

$$\mathcal{O}(n \log_{TcI})$$

for linear algorithms and

$$\mathcal{O}(n^{TcI})$$

for non-linear algorithms.

Summarily, the second case has been useful to provide data for evaluation of the applicability and efficacy of the algorithms, and to show that the artifacts are applicable to different business environments. To further evaluate the efficiency of the algorithms, the next sections proceeds to assess their performance.

9.6 Summary and Discussion

The chapter presented an abstraction of use case 2 as a basis for evaluating the composed algorithms. For further evaluation of capacity of the algorithms, the method evaluation model (MEM) was adopted.

The evaluation has revealed the appropriateness of the algorithms in detection of violations based on use case 2 compliance requirements and constraints. The algorithms have been evaluated through an informed argument of the use cases as a way to establish the extent to which they meet the design requirements. Using the MEM parameters, the experiments and results indicate that the efficacy of the artifacts are appropriate and feasible for verifying compliance of collaborative business processes with policy and regulatory constraints. However, a prototypical implementation is still necessary to enable a practical evaluation of our propositions especially in terms of time and space requirements.

Chapter 10

Conclusion and Future work

10.1 Introduction

The chapter summarises the work presented in this thesis, its findings and outputs. As a conclusion, the chapter recaps on the efforts taken to address the research question and aims that were stated in chapter 1 in respect to the research contributions. In section 10.2, general observations are made while in section 10.3 the contributions from this work are presented. Section 10.4 presents the limitations and open problems. Section 10.5 is the general conclusion . Lastly, section 10.6 sketches ideas and potential directions for future work.

10.2 General Observations

Compliance Management is a key business concept which when not given keen attention can lead to drastic repercussions for the organisation. The organisations have to ensure that their business processes align with the requirements specified in the policies and regulations.The assurance only comes when the business processes are verified for compliance. In Chapter 2 for related work, it was indicated that the subject of business process compliance has received several applications across the industry and academia. From the Figure 1.7 which situates the dimensions of the compliance spectrum in the state of art, most contributions have been made in the areas of compliance strategies and norms modelling. In our work, through a compliance verification approach composed of hybrid verification algorithms, we have contributed to the strategies domain. Additionally, we have also contributed to usability domain by presenting less complex mechanism and verification artifacts as the evaluation has shown. In the next section, the specific contributions from the thesis are presented in more details.

10.3 Contributions

The work presented in this thesis aimed at supporting end users to verify collaborative business processes for compliance with policy and regulatory requirements. The outcome is a compliancy verification approach supporting elicitation, specification and analysis of policy and regulatory requirements, their translation into formal constraints that are verifiable with collaborative business processes for compliance. Towards achieving this goal, a number of contributions were made, these are discussed in relation to the objectives that guided the study.

End users like the compliance officers are not known to be experts in business process modelling. The kind of support they need involves translation of the compliance requirements into formal constraints in a simple comprehensible language that they can use so that they can fully participate in the compliance process. Existing tools or approaches require strong technical and mathematical skills that ordinary users may lack. The consequential contribution to this requirement was a compliance mechanism in form of a language based on description logic, which is close to natural language yet expressive enough to support specification and reasoning about constraints and process behaviour. To make the mechanism self-independent, we integrated basic constructs from temporal logic in form of operators and quantifiers to support and facilitate reasoning over the constraints and support verification.

The following contributions are realised in line with the objectives stated in Chapter 1.

Objective: To support the elicitation and translation of compliance requirements from source documents into compliancy constraints.

Contribution 1: The contribution based on the above objective was realised with a mechanism that end users can use to elicit and express relevant policy and regulatory requirements from source documents. In this aspect, the application of DL based constraint expression language was composed and presented with illustrations based on concrete business use cases. In comparison, the constraint definition languages presented in the related work (Chapter 2) are associated with difficulty in application by requiring expertise or being specific to a given domain e.g CRL [46], PENELOPE [56, 57, 55], FCL [131, 130] and SecBPMN [132]. Our mechanism is in form of a language that whose syntactical and semantical structure are close to natural language making is easy and ideal for application by ordinary users.

Objective:To demonstrate the application of simulation and analysis as a technique to support policy and regulatory variations and impact assessment.

Contribution 2: From objective 2, a contribution in form of application of Simulation analysis technique was demonstrated in Chapter 4 Section 4.6.1 to analyze and assess the impact of change in policy and regulatory requirements over key performance indicators in the business process. To arrive at realistic projections, we based the analysis on an industry based use case whose policies were varied according to different requirements specified in various scenarios to determine the best, average or worst cases to support decision making. Additionally, simulation supported the generation of traces that were useful to aid verification especially where real data is not available. Traditionally, traces are mined from information system or business process management system logs through process mining. This contribution reveals the role of Simulation in analysis and generation of process instances and traces to facilitate design time verification before actual implementation of system.

Objective: Design a compliance verification approach for supporting compliance verification of collaborative business processes.

Contribution 3: An integrated compliance verification approach resulted as a contribution from objective 3. This contribution in the thesis is presented in chapter 7 as Figure 3.5. The approach is a comprehensive integrated road map towards achievement of compliance verification for collaborative business processes. Key of the components in the approach are various constraint specific verification algorithms through which compliance of the business processes with policy and regulatory requirements is checked. Besides constraint specific algorithms, an overall compliance verification checking algorithm is presented as algorithm (12). To demonstrate the design and empirical applicability of the algorithms, two industry based collaborative business processes are used, i.e. pick and pack case and the car insurance trading process for design and evaluation respectively. Based on the outcome, the efficacy, efficiency and performance of the algorithms are reported in chapter 10. However, further evaluation is recommended as future work based on practical implementation of the prototype.

Objective: Apply process driven authorisation and access control (PDAC), a novel mechanism for implementing privacy and authentication

Contribution 4: Based on the above objective, a pair of architectures are composed and presented for practical implementation of both PDAC - a process driven authorisation access control mechanism and another for a service verification languages. The PDAC is a novel access control mechanism for authorising access based on purpose and time required to access data. PDAC leverages traditional access control mechanisms based on task and roles as discussed in related work in Chapter 2. The contribution of these architectures forms a basis for future practical implementation of the compliance verification approach. The implementation of prototypes based on the architectures would facilitate practical evaluation of the artifacts using large data sets to assess the efficacy and efficiency of the artifacts.

Contribution 5: Based on Figure 1.6 in section 1.5, it was noted that an organisation is made of different strategies and compliance management remains at a core focal point to ensure that each of these strategies achieves compliance requirements. For instance, the software industry produces software that must adhere to the software standards. The Project management industry is regulated by various bodies like PRINCE which specify rules that must be complied with. The same applies for change management strategy. All these strategies have life cycles through which they are managed. At each phase of the life cycle, regulatory compliancy ensures adherence to the relevant laws, rules, standards and other forms of regulations through verification. In this thesis, we have supported management of compliance requirements for the business process management life cycle as an organisation strategy. However, we have put forwards generic concepts, mechanisms and artifacts that are not only limited to the BPM life cycle but can be applied in managing compliance requirements for other organisational strategies.

Dissemination of the Research outcomes:

The research outcomes have been disseminated in several presentations. For example; at conferences, publications in conference proceedings, journal and key milestones in the EU H2020 FIRST project. It is hoped that these contributions to the body of literature will further the subject of compliance verification in relation to the future research directions.

10.4 Limitations

Some limitations were encountered especially due to the time frame in which the work was to be accomplished. Regarding subject based limitations, some concepts have not been tackled. One of them is loops. looping is a constraint common to almost all business processes. The proposed mechanisms do not fully address verification concerns related to looping structures. We contend that loops are potential causes of violations if not designed or verified. The concept of verifying constraints relating to loops forms a proposition for a future direction of this research. A further limitation is that a prototype has not been realised by this time which has limited the practical evaluations to fully demonstrate our propositions of the language and algorithms. Despite the unimplemented prototype, the specification language and the algorithms have been validated and

evaluated using two different industry use cases. This is a considerable attestation of application and value of the thesis outcomes.
The evaluation of the algorithms revealed that they are expressive enough to independently support end users to verify compliance of collaborative business processes.

10.5 Conclusion

The work presented in this thesis was set out to support compliance verification for collaborative business processes with a focus on non-expert end users who are the subject matter experts yet not competent in modelling and verification. From the related work it was noted that despite the existing research in academia and industry, a definite solution for compliance verification of collaborative business processes is still lacking because they present unique characteristics. These characteristics as elaborated in the related work, they present specific verification requirements against the various dynamic policies and regulations. These requirements must be complied with by the business process.

As a contribution from the thesis, we proposed a compliance management approach informed by a set of requirements and objectives which have been addressed by the study.
Requirement 1: Compose a requirements elicitation and definition mechanism.
To address this requirement, a requirements definition mechanism was composed based on description language and linear temporal logic (LTL). The mechanism supports end users to elicit compliance requirements from source documents of policies and regulations. More over, the compliance requirements are translated into formal semantics and expressed as formal constraints. The mechanism is easy to use for ordinary users due to its closeness to natural language.
Requirement 2: Supporting compliance verification.
Addressing this requirement involved composition of various category based sub algorithms and algorithms to support model checking at various levels; during process design, runtime and compliance auditing. The algorithms have been validated and evaluated using two different industry use cases. The outcome shows their efficacy, efficiency and applicability.
Requirement 3: Providing Intelligible Feedback to the end user
One of the key limitations noted from the related is feed back that is technical and incomprehensible to end users. The algorithms put forward provide feedback to the user in an easy and understandable form pointing to the source of compliance violation. This further illustrates compliance verification support for non-expert end users.

Besides the fulfilled requirements, the thesis is based on a technical foundation as shown in the conceptual framework. This involved study and critical analysis of the existing state of the art in compliance verification to support the propositions put forward. This way, the thesis contributes to the existing knowledge in the state of the art.

10.6 Potential Future Research Directions

The projected direction of this research points to four key areas, i.e. addressing the concept of time based violations, addressing loops as a source of process violations, Proving more soundness of proposed language based specifications, and implementing a practical prototype based on the designed architectures.

- Time based verification: The concept of temporal requirements has been addressed but not to a logical conclusion required for collaborative business processes. For example, temporal violations based on relative or fixed duration. Time based verification will be further addressed in the future.

- Loops: Loops are common for all business processes. Different forms of loops based on XOR or optional loops impose different restrictions on process behaviour. During execution, loops introduce new states which may be a source of violations in the business process if not verified, thus leading to eventual non-conformance of the process. As a future direction, verification of loop based violations in collaborative business processes will be addressed.

- The proposed language based on description logic and linear temporal logic needs further engineering to prove the soundness and completeness of the logic formulae. This will be focused on in the future.

- Lastly, design and implementation of a prototype based on the composed algorithms and architectures forms the other future direction. The prototype will facilitate a practical evaluation of the proposed theory in form of a compliance verification approach. We plan to implement the prototype with supporting databases enhanced with running real life data to enable performance evaluation based on experiments. The implementation may involve refinements of the compliance approach, and /or the algorithms. More over, the implementation will follow the architectures proposed in Figures 10.1 and 7.6. In the next section the details of the proposed architecture for the proposed prototype are discussed.

10.7 Prototype Implementation

To implement an environment that would facilitate practical validation and resolve compliance issues in collaborative business processes to achieve compliant business processes calls for a practical prototype implementation. The prototype would be based on the architecture presented in Figure 10.1, composed of interacting modules that utilise both policies and regulations, and business process models to enforce compliance verification. The architecture illustrates the three main modules and their interaction:

Module 1 - Model and Specify: The module is composed of 3 other sub modules that interact to achieve compliancy verification. The interaction is as follows;

Sub module (i) is constraints formulation whose goal is to support compliance requirements and constraints specification. It facilitates the elicitation of policy and regulatory requirements and their formalisation into compliance constraints.

Sub module (ii) supports the mapping of formal constraints with the policy and regulatory constraints while sun module (iii) is dedicated to supporting verification of the business process's compliance with the constraints. The database stores both process models and constraints. It also provides the inputs for constraints and models during verification as well as a storage for verification outcomes and feedback.

Module 2 - Verification Service: The module is a service invoked during verification by sub module (iii) via an API. It is encapsulated to be independent of module 1. This implies that the verification service is not application specific or domain specific, whereby various verification requirements can be supported. The service is composed of a verification engine which employs described techniques and mechanism, i.e. the simulation technique (discussed in Chapter 4), compliance verification algorithms (presented in Chapter 7) and the process driven access control and authorisation mechanism (discussed in section 7.7).

Module 3 - Feedback and Reporting: The module provides feedback to the end user about the compliancy or the violations detected during verification. The feedback should be intelligible and in a format comprehensible for non-expert end users.

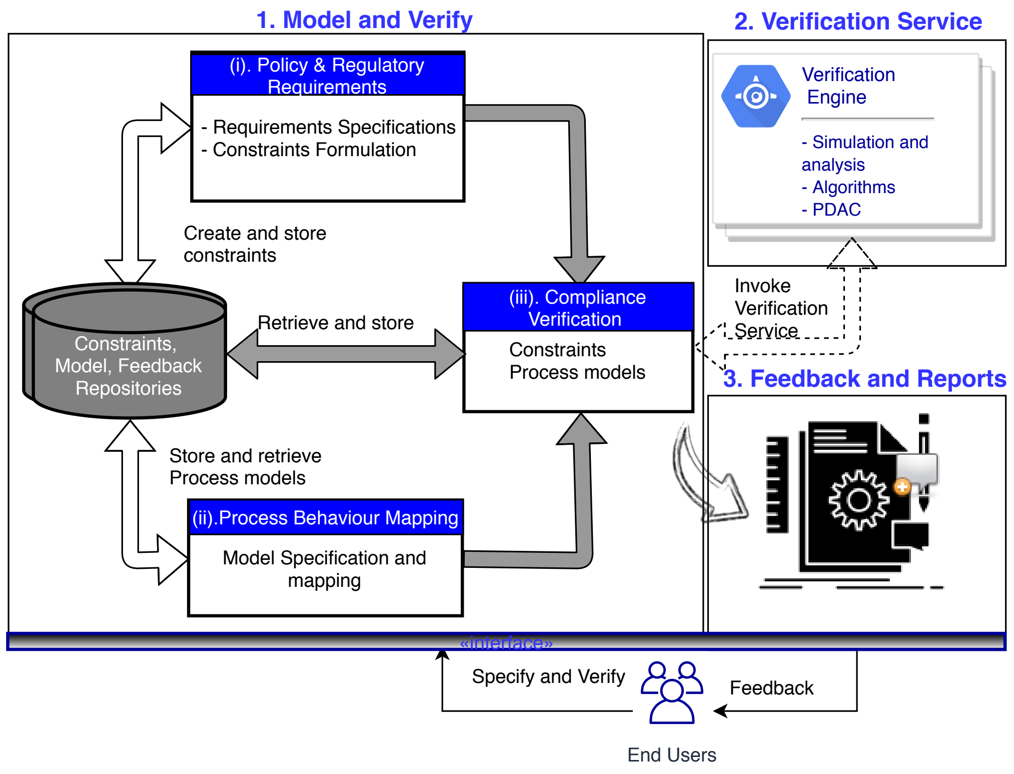
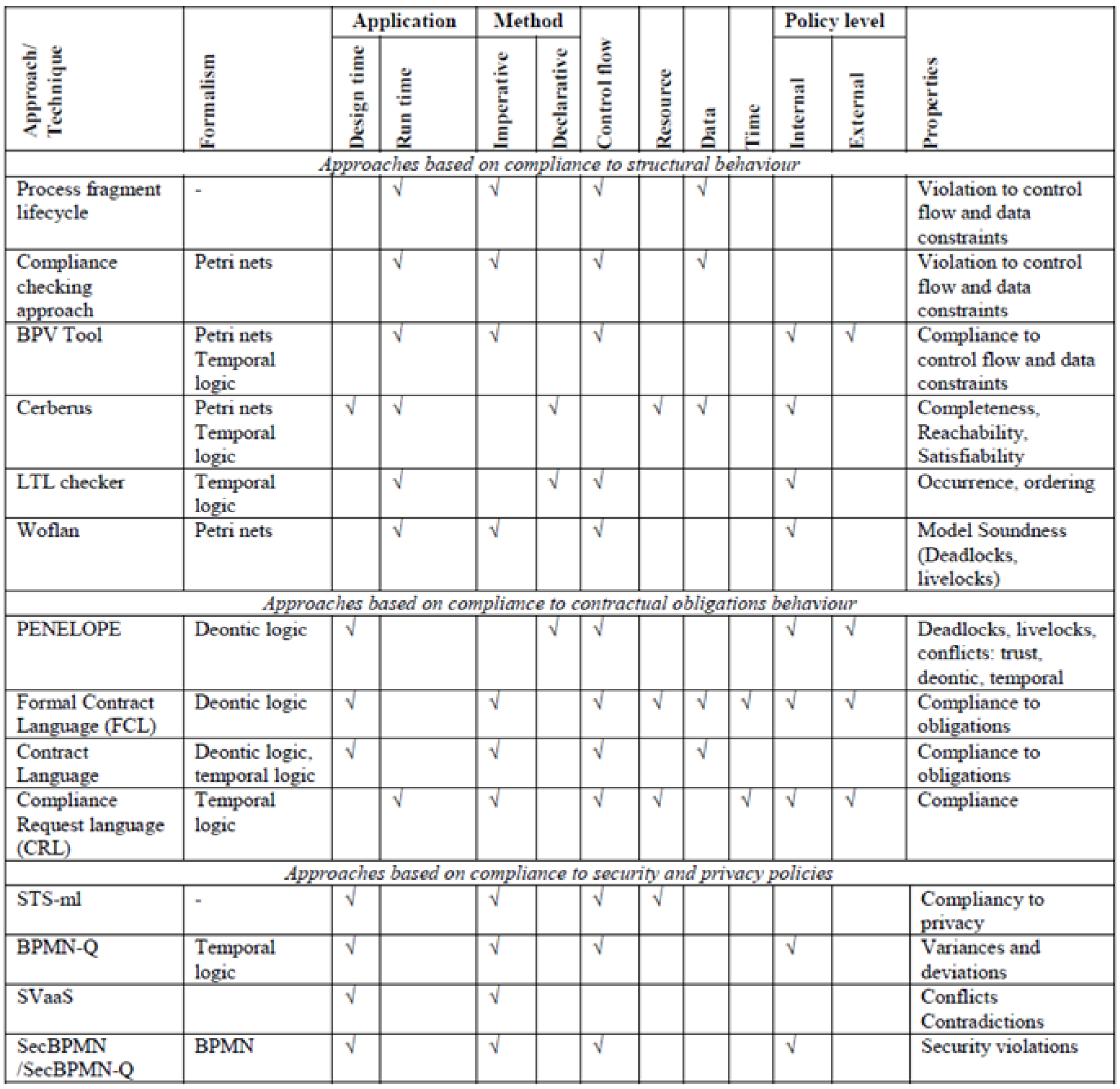

1. Model and Verify
2. Verification Service
(i). Policy & Regulatory Requirements
- Requirements Specifications
- Constraints Formulation
Verification Engine
- Simulation and analysis
- Algorithms
- PDAC
Create and store constraints
Invoke Verification Service
(iii). Compliance Verification
Constraints
Process models
Retrieve and store
Constraints, Model, Feedback Repositories
3. Feedback and Reports
Store and retrieve Process models
(ii).Process Behaviour Mapping
Model Specification and mapping
«interface»
Specify and Verify
Feedback
End Users

Figure 10.1: Verification Prototype Architecture

Appendix A

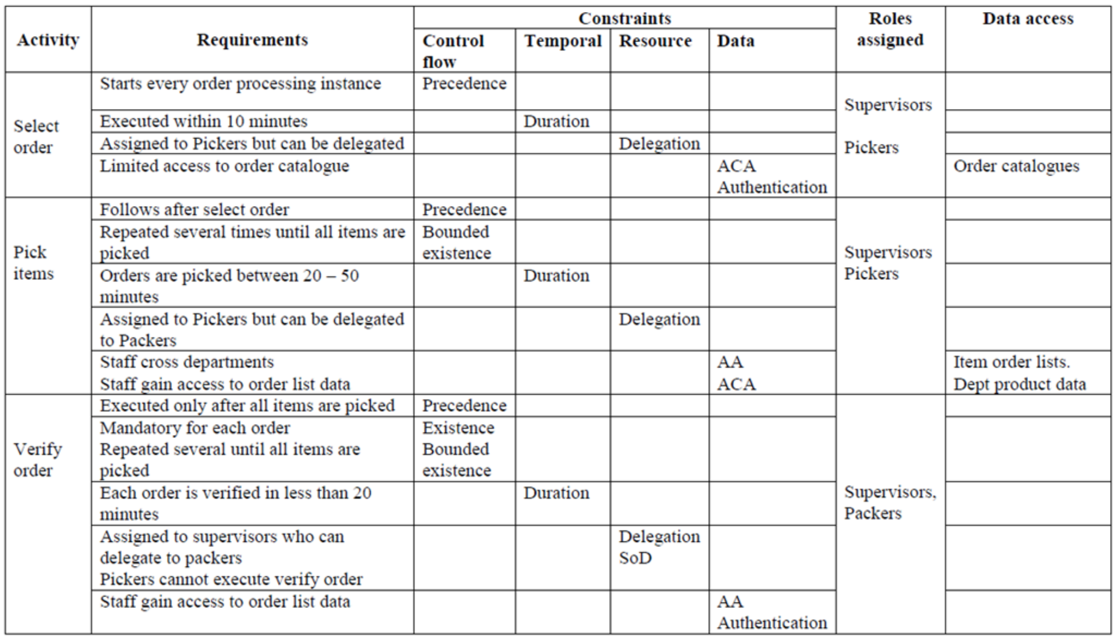
A.1 Summary of Compliance Verification Approaches



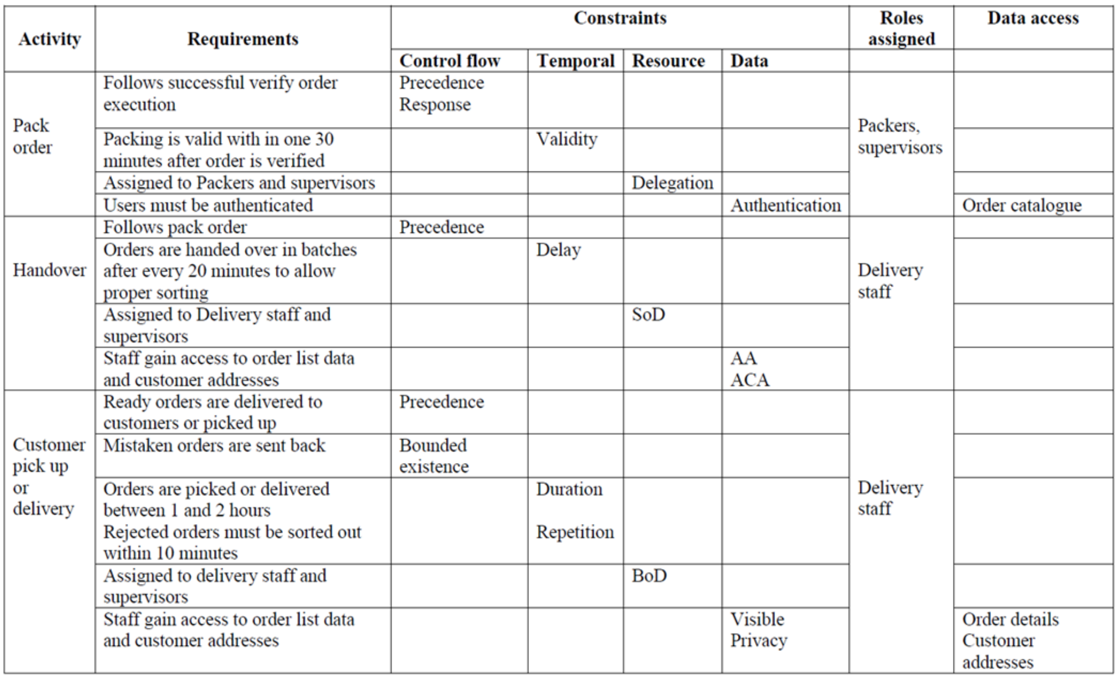
Approach/ Technique	Formalism	Application		Method		Control flow	Resource	Data	Time	Policy level		Properties
		Design time	Run time	Imperative	Declarative					Internal	External	
Approaches based on compliance to structural behaviour												
Process fragment lifecycle	-		√	√		√		√				Violation to control flow and data constraints
Compliance checking approach	Petri nets		√	√		√		√				Violation to control flow and data constraints
BPV Tool	Petri nets Temporal logic		√	√		√				√	√	Compliance to control flow and data constraints
Cerberus	Petri nets Temporal logic	√	√		√		√	√		√		Completeness, Reachability, Satisfiability
LTL checker	Temporal logic		√		√	√				√		Occurrence, ordering
Woflan	Petri nets		√	√		√				√		Model Soundness (Deadlocks, livelocks)
Approaches based on compliance to contractual obligations behaviour												
PENELOPE	Deontic logic	√			√	√				√	√	Deadlocks, livelocks, conflicts: trust, deontic, temporal
Formal Contract Language (FCL)	Deontic logic	√		√		√	√	√	√	√	√	Compliance to obligations
Contract Language	Deontic logic, temporal logic	√		√		√		√				Compliance to obligations
Compliance Request language (CRL)	Temporal logic		√	√		√	√		√	√	√	Compliance
Approaches based on compliance to security and privacy policies												
STS-ml	-	√		√		√	√					Compliancy to privacy
BPMN-Q	Temporal logic	√		√		√				√		Variances and deviations
SVaaS		√		√								Conflicts Contradictions
SecBPMN /SecBPMN-Q	BPMN	√		√		√				√		Security violations

A.2 Requirements and Constraints for pick and pack exemplar case - Application

Requirements and Constraint Lists



Activity	Requirements	Constraints				Roles assigned	Data access
		Control flow	Temporal	Resource	Data		
Select order	Starts every order processing instance	Precedence				Supervisors Pickers	
	Executed within 10 minutes		Duration				
	Assigned to Pickers but can be delegated			Delegation			
	Limited access to order catalogue				ACA Authentication		Order catalogues
Pick items	Follows after select order	Precedence				Supervisors Pickers	
	Repeated several times until all items are picked	Bounded existence					
	Orders are picked between 20 – 50 minutes		Duration				
	Assigned to Pickers but can be delegated to Packers			Delegation			
	Staff cross departments Staff gain access to order list data				AA ACA		Item order lists. Dept product data
Verify order	Executed only after all items are picked	Precedence				Supervisors, Packers	
	Mandatory for each order Repeated several until all items are picked	Existence Bounded existence					
	Each order is verified in less than 20 minutes		Duration				
	Assigned to supervisors who can delegate to packers Pickers cannot execute verify order			Delegation SoD			
	Staff gain access to order list data				AA Authentication		



Activity	Requirements	Constraints				Roles assigned	Data access
		Control flow	Temporal	Resource	Data		
Pack order	Follows successful verify order execution	Precedence Response				Packers, supervisors	
	Packing is valid with in one 30 minutes after order is verified		Validity				
	Assigned to Packers and supervisors			Delegation			
	Users must be authenticated				Authentication		Order catalogue
Handover	Follows pack order	Precedence				Delivery staff	
	Orders are handed over in batches after every 20 minutes to allow proper sorting		Delay				
	Assigned to Delivery staff and supervisors			SoD			
	Staff gain access to order list data and customer addresses				AA ACA		
Customer pick up or delivery	Ready orders are delivered to customers or picked up	Precedence				Delivery staff	
	Mistaken orders are sent back	Bounded existence					
	Orders are picked or delivered between 1 and 2 hours Rejected orders must be sorted out within 10 minutes		Duration Repetition				
	Assigned to delivery staff and supervisors			BoD			
	Staff gain access to order list data and customer addresses				Visible Privacy		Order details Customer addresses

Bibliography

[1] W M P Van Der Aalst. “Verification of Workflow Nets”. In: (1997).

[2] W M P Van Der Aalst, A H M Hofstede, and M Weske. “Business Process Management: A Survey”. In: *Business Process Management* (2003), pp. 1–12. ISSN: 03029743. DOI: 10.1007/3-540-44895-0.

[3] W M P van der Aalst. “Business Process Management : A Comprehensive Survey”. In: *ISRN Software Engineering* 2013 (2013), pp. 1–37. ISSN: 2090-7680. DOI: http://dxdoi.org/10.1155/2013/507984.

[4] W. M. P. van der Aalst, H. T. de Beer, and B. F. van Dongen. “Process Mining and Verification of Properties: An Approach Based on Temporal Logic”. In: (2005), pp. 130–147. ISSN: 03029743. DOI: 10.1007/11575771_11. URL: http://link.springer.com/10.1007/11575771%7B%5C_%7D11.

[5] Wil M P Van Der Aalst, Arthur H M Hofstede, and Mathias Weske. “Business Process Management : A Survey”. In: (2003), pp. 1–12.

[6] Wil M P van der Aalst. “Business Process Management Demystified: A Tutorial on Models, Systems and Standards for Workflow Management.” In: *Business process management demystified: A tutorial on models, systems and standards for workflow management* 3098 (2004), pp. 1–65. ISSN: 0302-9743. DOI: 10.1007/978-3-540-27755-2_1.

[7] Wil M P van der Aalst et al. “Design and Implementation of the YAWL System”. In: *Advanced Information Systems Engineering* (2004), pp. 281–305. ISSN: 03029743. DOI: 10.1007/978-3-540-25975-6_12. URL: http://www.springerlink.com/content/cpa194xbmauduuwn.

[8] Wil MP van der Aalst and Arthur HM ter Hofstede. “YAWL: Yet another workflow language (revised version)”. In: *Queensland Univ. Technol., Brisbane, Australia, QUT Tech. Rep., FIT-TR-2003-04* (2003).

[9] Wil van der Aalst. “Loosely coupled interorganizational workflows: modeling and analyzing workflows crossing organizational boundaries”. In: *Information & Management* 37.2 (2000), pp. 67–75. ISSN: 03787206. DOI: http://dx.doi.org/10.1016/S0378-7206(99)00038-5. URL: http://www.sciencedirect.com/science/article/B6VD0-3YJ9Y2V-2/2/d9c28a0dfa2816dcd7f419de6a56d7cf%7B%5C%%7D5Cnhttp://www.sciencedirect.com/science/article/pii/S0378720699000385.

[10] Wil van der Aalst and Kees van Hee. “Workflow Management”. In: (2004).

[11] Martín Abadi, Leslie Lamport, and Stephan Merz. “A TLA solution to the RPC-memory specification problem”. In: 1996, pp. 21–66. ISBN: 3-540-61984-4. DOI: 10.1007/BFb0024426. URL: http://link.springer.com/10.1007/BFb0024426.

[12] Marco Aiello, Pavel Bulanov, and Heerko Groefsema. “Requirements and tools for variability management”. In: *Proceedings - International Computer Software and Applications Conference* July (2010), pp. 245–250. ISSN: 07303157. DOI: 10.1109/COMPSACW.2010.50.

[13] A. Alshehri and R. Sandhu. “Access Control Models for Virtual Object Communication in Cloud-Enabled IoT”. In: *2017 IEEE International Conference on Information Reuse and Integration (IRI).* 2017.

[14] Rajeev Alur, Thomas A Henzinger, and Pei-hsin Ho. “Automatic Sybolic Verification of Embedded Systems”. In: *IEEE Transactions on Software Engineering* 22.3 (1996), pp. 2–11.

[15] Ahmed Awad. “BPMN-Q: A Language to Query Business Processes”. In: *In Proceedings of EMISA’07* (2007), pp. 115–128.

[16] Ahmed Mamoud Awad. “A C OMPLIANCE M ANAGEMENT F RAMEWORK FOR B USINESS P ROCESS M ODELS”. In: *PhD Thesis* (2010).

[17] Ahmed Awad, Gero Decker, and Mathias Weske. “Efficient Compliance Checking Using BPMN-Q and Temporal Logic”. In: (2008).

[18] Christel Baier and Joost-Pieter Katoen. *Principles Of Model Checking*. Vol. 950. 2008, pp. I–XVII, 1–975. ISBN: 9780262026499. DOI: 10.1093/comjnl/bxp025. URL: http://mitpress.mit.edu/books/principles-model-checking.

[19] Basel. *The Basel iii Accord*. 2018. URL: https://www.basel-iii-accord.com/ (visited on).

[20] H T De Beer and PCW van de Brand. "The LTL Checker Plugins - a (reference) manual". In: (2007).

[21] Johan Bengtsson et al. "in 1995". In: December (1996).

[22] Ferrari Elena Atluri Vijayalakshmi Bertino Elisa. "Flexible model supporting the specification and enforcement of role-based authorizations in workflow management systems". In: *Proceedings of the ACM Workshop on Role-Based Access Control* (1997), pp. 1–12. URL: http://www.scopus.com/inward/record.url?eid=2-s2.0-0031382313%7B%5C&%7DpartnerID=40%7B%5C&%7Dmd5=.

[23] Elisa Bertino, Elena Ferrari, and Vijay Atluri. "The specification and enforcement of authorization constraints in workflow management systems". In: *ACM Transactions on Information and System Security* 2.1 (1999), pp. 65–104. ISSN: 10949224. DOI: 10.1145/300830.300837. URL: http://portal.acm.org/citation.cfm?doid=300830.300837.

[24] Elisa Bertino et al. "An access control model supporting periodicity constraints and temporal reasoning". In: *ACM Transactions on Database Systems (TODS)* 23.3 (1998), p. 231. ISSN: 0362-5915. DOI: http://doi.acm.org/10.1145/293910.293151. URL: http://portal.acm.org/citation.cfm?doid=293910.293151.

[25] Khoutir Bouchbout, Jacky Akoka, and Zaia Alimazighi. "An MDA-based framework for collaborative business process modelling". In: *Business Process Management Journal* 18.6 (2012), pp. 919–948. ISSN: 1463-7154. DOI: 10.1108/14637151211283357. URL: http://www.emeraldinsight.com/10.1108/14637151211283357%7B%5C%%7D5Cnhttp://www.emeraldinsight.com/doi/abs/10.1108/14637151211283357.

[26] Khoutir Bouchbout and Zaia Alimazighi. "Inter-organizational business processes modelling framework". In: *ADBIS (2)*. Citeseer. 2011, pp. 45–54.

[27] Khoutir Bouchbout and Zaia Alimazighi. "Inter-organizational business processes modelling framework". In: *CEUR Workshop Proceedings*. Vol. 789. 2011, pp. 45–54.

[28] BPI. *BPM Glossary*. 2019. URL: https://www.businessprocessglossary.com/ (visited on 03/16/2019).

[29] A Bryman and E. Bell. *Business Research Methods*. 2015, p. 27.

[30] Cristina Cabanillas et al. "Specification and automated design-time analysis of the business process human resource perspective". In: *Information Systems* 52 (2015), pp. 55–82. ISSN: 03064379. DOI: 10.1016/j.is.2015.03.002. URL: http://dx.doi.org/10.1016/j.is.2015.03.002.

[31] Cristina Cabanillas et al. "Specification and automated design-time analysis of the business process human resource perspective". In: *Information Systems* 52 (2015), pp. 55–82. ISSN: 03064379. DOI: 10.1016/j.is.2015.03.002. URL: http://dx.doi.org/10.1016/j.is.2015.03.002.

[32] II Carson and S John. "Introduction to modeling and simulation". In: *Proceedings of the 37th conference on Winter simulation*. Winter Simulation Conference. 2005, pp. 16–23.

[33] Yolanda Carson and Anu Maria. "Simulation optimization: methods and applications". In: *Proceedings of the 29th conference on Winter simulation*. IEEE Computer Society. 1997, pp. 118–126.

[34] Fabio Casati and Angela Discenza. "Supporting workflow cooperation within and across organizations". In: *Proceedings of the 2000 ACM symposium on Applied computing* c (2000), pp. 196–202. DOI: 10.1145/335603.335742.

[35] Saoussen Cheikhrouhou, Slim Kallel, and Mohamed Jmaiel. "Toward a verification of time-centric business process models". In: *Proceedings of the Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE* (2014), pp. 326–331. ISSN: 15244547. DOI: 10.1109/WETICE.2014.75.

[36] Yulia Cherdantseva and Jeremy Hilton. "A Reference Model of Information Assurance & Security". In: *International Conference on Availability, Reliability and*

Security (2013), pp. 546–555. DOI: 10.1109/ARES.2013.72. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6657288.

[37] Alessandro Cimatti, Edmund Clarke, and E Giunchiglia. "Nusmv 2: An open-source tool for symbolic model checking". In: *Computer Aided Verification* 2404 (2002), pp. 359–364. ISSN: 16113349. DOI: 10.1007/3-540-45657-0_29. URL: http://link.springer.com/chapter/10.1007/3-540-45657-0%7B%5C_%7D29.

[38] E. M. Clarke, E. a. Emerson, and a. P. Sistla. "Automatic verification of finite-state concurrent systems using temporal logic specifications". In: *ACM Transactions on Programming Languages and Systems* 8.2 (1986), pp. 244–263. ISSN: 01640925. DOI: 10.1145/5397.5399.

[39] Luca Compagna, Pierre Guilleminot, and Achim D. Brucker. "Business process compliance via security validation as a service". In: *Proceedings - IEEE 6th International Conference on Software Testing, Verification and Validation, ICST 2013* (2013), pp. 455–462. ISSN: 2159-4848. DOI: 10.1109/ICST.2013.63.

[40] Jason Crampton and Gregory Gutin. "Constraint Expressions and Workflow Satisfiability". In: *Proceedings of the 18th ACM symposium on Access control models and technologies. ACM* (2013), pp. 73–84. DOI: 10.1145/2462410.2462419. arXiv: 1301.3402. URL: http://arxiv.org/abs/1301.3402.

[41] CTMfile. *How much will GDPR cost your company?* 2017.

[42] T H Davenport. "Process Innovation". In: *Harvard Business School Press* (1993).

[43] B. van Dongen et al. "The ProM framework: A new era in process mining tool support". In: *Application and Theory of Petri Nets 2005* 3536 (2005), pp. 444–454. ISSN: 03029743. DOI: 10.1007/11494744_25. URL: http://wwwis.win.tue.nl/%7B~%7Dwvdaalst/publications/p264.pdf.

[44] Marlon Dumas et al. *Fundamentals of Business Process Management.* 2012. ISBN: 9783642331428.

[45] Matthew B. Dwyer, S. George Avrunin, and James C. Corbett. "Property Specification Patterns for Finite-State Verification". In: *FMSP '98 Proceedings of the second workshop on Formal methods in software practice* (1998), pp. 1–7.

[46] Amal Elgammal et al. "Formalizing and appling compliance patterns for business process compliance". In: *Software and Systems Modeling* 15.1 (2016), pp. 119–146. ISSN: 16191374. DOI: 10.1007/s10270-014-0395-3.

[47] Yliès Falcone, Klaus Havelung, and Giles Reger. "A Tutorial on Runtime Verification". In: *Engineering Dependable Software Systems* 34 (2013), pp. 141–175. DOI: 10.3233/978-1-61499-207-3-141. URL: http://hal.inria.fr/hal-00853727.

[48] Walid Fdhila, Stefanie Rinderle-Ma, and Manfred Reichert. "Change Propagation in Collaborative Processes Scenarios". In: (2013). DOI: 10.4108/icst.collaboratecom.2012.250408.

[49] Walid Fdhila et al. "Change and Compliance in Collaborative Processes". In: *Proceedings - 2015 IEEE International Conference on Services Computing, SCC 2015* (2015), pp. 162–169. DOI: 10.1109/SCC.2015.31.

[50] David F Ferraiolo et al. "Proposed NIST standard for role-based access control". In: *ACM Transactions on Information and System Security (TISSEC)* 4.3 (2001), pp. 224–274.

[51] Frederick K. Frantz. "A taxonomy of model abstraction techniques". In: *Proceedings of the 27th conference on Winter simulation* (1995), pp. 1413–1420. DOI: 10.1145/224401.224834.

[52] El Gammal. "Towards a comprehensive framework for business process compliance". In: (2014).

[53] Dimitra Giannakopoulou and Klaus Havelund. "Automata-based verification of temporal properties on running programs". In: *Proceedings 16th Annual International Conference on Automated Software Engineering (ASE 2001)* August (2001), pp. 412–416. ISSN: 1527-1366. DOI: 10.1109/ASE.2001.989841.

[54] Barney Glaser. "Grounded theory methodology". In: *Introducing Qualitative Research in Psychology* (2013), pp. 69–82.

[55] Stijn Goedertier. "Declarative Techniques for Modeling and Mining Business Processes." In: 284 (2008), p. 248. URL: https://lirias.kuleuven.be/handle/1979/1908.

[56] Stijn Goedertier and Jan Vanthienen. "Designing Compliant Business Processes with Obligations and Permissions". In: *BPM 2006 International Workshops, BPD, BPI, ENEI, GPWW, DPM, semantics4ws, Vienna, Austria, September 4-7, 2006.* (2006), pp. 5–14. ISSN: 03029743. DOI: 10.1007/11837862_2. URL: http://link.springer.com/10.1007%7B%5C%%7D252F11837862%7B%5C_%7D2.

[57] Goedertier and Vanthienen. "Compliant and Flexible Business Processes with\nBusiness Rules". In: *Bpmds* January (2007), pp. 94–103. ISSN: 978-2-87037-525-9. URL: https://lirias.kuleuven.be/handle/123456789/103560%7B%5C%%7D255Cnhttp://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-236/paper3.pdf.

[58] F. Gottschalk et al. "Protos2CPN: Using colored Petri nets for configuring and testing business processes". In: *International Journal on Software Tools for Technology Transfer* 10.1 (2008), pp. 95–110. ISSN: 14332779. DOI: 10.1007/s10009-007-0055-9.

[59] Guido Governatori and Antonino Rotolo. "How do agents comply with norms?" In: *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 03.* IEEE Computer Society. 2009, pp. 488–491.

[60] Heerko Groefsema. *Business Process Variability A:A Study into Process Management and Verification.* 2016. ISBN: 9789036792370.

[61] Heerko Groefsema and Doina Bucur. "A Survey of Formal Business Process Verification: From Soundness to Variability". In: *Proceedings of the Third International Symposium on Business Modeling and Software Design* (2013), pp. 198–203.

[62] Heerko Groefsema, Pavel Bulanov, and Marco Aiello. "Declarative Enhancement Framework for Business Processes". In: 7084 (2011). DOI: 10.1007/978-3-642-25535-9.

[63] Heerko Groefsema, Pavel Bulanov, and Marco Aiello. "Imperative versus Declarative Process Variability : Why Choose ?" In: (2012), pp. 1–52.

[64] Alena Hallerbach, Thomas Bauer, and Manfred Reichert. "Managing process variants in the process life cycle". In: (2008), pp. 154–161.

[65] Joseph Y Halpern and Vicky Weissman. “Using first-order logic to reason about policies”. In: *ACM Transactions on Information and System Security (TISSEC)* 11.4 (2008), p. 21.

[66] Michael Hammer and James Champy. “R EENGINEERING T HE C ORPORATION A Manifesto For Business Revolution”. In: (1993).

[67] Mustafa Hashmi et al. “Are we done with business process compliance: state of the art and challenges ahead”. In: *Knowledge and Information Systems* (2018), pp. 1–55.

[68] Klaus Havelund and Grigore Roşu. “Synthesizing Monitors for Safety Properties”. In: *CEUR Workshop Proceedings* 1542.9 (2002), pp. 342–356. ISSN: 16130073. DOI: 10.1007/3-540-46002-0_24. arXiv: arXiv:1011.1669v3. URL: http://link.springer.com/10.1007/3-540-46002-0%7B%5C_%7D24.

[69] Klaus Havelund, Grigore Rosu, and Peter Norvig. “Testing linear temporal logic formulae on finite execution traces”. In: (2001).

[70] Thomas A. Henzinger, Benjamin Horowitz, and Rupak Majumdar. “Beyond HyTech :” in: (1999), pp. 89–95.

[71] Thomas A Henzinger and Howard Wong-toi. “HyTech : A Model Checker for Hybrid Systems 1 Introduction”. In: *International Journal on Software Tools for Technology Transfer (STTT)* 1.1997 (1997), pp. 110–122. ISSN: 1433-2779. DOI: 10.1007/s100090050008.

[72] Alan R Hevner. “A three cycle view of design science research”. In: *Scandinavian journal of information systems* 19.2 (2007), p. 4.

[73] Alan R Hevner, South Florida, and Salvatore T March. “Cycle”. In: (1996), pp. 111–113.

[74] Alan R Hevner et al. “Research Essay Design Science in Information”. In: *MIS Quarterly* 28.1 (2004), pp. 75–105.

[75] Gerard Holzmann. “The Design and Validation of the CLASS.pdf”. In: March (2017).

[76] Gerard J. Holzmann. "The model checker SPIN". In: *IEEE Transactions on Software Engineering* 23.5 (1997), pp. 279–295. ISSN: 00985589. DOI: 10.1109/32.588521.

[77] Gerard J. Holzmann, Patrice Godefroid, and Didier Pirottin. "Coverage preserving reduction strategies for reachability analysis". In: 6 (2013), pp. 349–363. URL: https://books.google.com/books?hl=en%7B%5C&%7Dlr=%7B%5C&%7Did=Q1EvBQAAQBAJ%7B%5C&%7Doi=fnd%7B%5C&%7Dpg=PA349%7B%5C&%7Ddq=%7B%5C%%7D22establish+the+correctness+of+systems+of+interacting+concurrent+processes+by+an%7B%5C%%7D22+%7B%5C%%7D22Sections+3+and+4+discuss+the+foundation+for+a+partial+order+semantics%7B%5C%%7D22+%7B%5C%%7D22the+actions+aff.

[78] Vincent C. Hu, D. Richard Kuhn, and David F. Ferraiolo. "Attribute-based access control". In: *Computer* 48.2 (2015), pp. 85–88. ISSN: 00189162. DOI: 10.1109/MC.2015.33.

[79] Vincent C. Hu et al. "Guide to Attribute Based Access Control (ABAC) Definition and Considerations". In: (2014). DOI: 10.6028/NIST.SP.800-162. URL: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-162.pdf.

[80] Zhengxing Huang, Xudong Lu, and Huilong Duan. "Mining association rules to support resource allocation in business process management". In: 38 (2011), pp. 9483–9490. DOI: 10.1016/j.eswa.2011.01.146.

[81] Kurt Jensen, Lars Michael Kristensen, and Lisa Wells. "Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems". In: *International Journal on Software Tools for Technology Transfer* 9.3-4 (2007), pp. 213–254. ISSN: 14332779. DOI: 10.1007/s10009-007-0038-x.

[82] Xin Jin, Ram Krishnan, and Ravi Sandhu. "A unified attribute-based access control model covering DAC, MAC and RBAC". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 7371 LNCS. 2012, pp. 41–55. ISBN: 9783642315398. DOI: 10.1007/978-3-642-31540-4_4.

[83] Craig Johnson. "Enron ' s Ethical Collapse : Lessons for Leadership Educators". In: *Journal of Leadership Education* 2.1 (2003), pp. 45–56. ISSN: 1552-9045.

[84] Kasse JP et al. "Process Driven Access Control and Authorisation Approach". In: (2019).

[85] Masaya Kadono, Tatsuhiro Tsuchiya, and Tohru Kikuno. "Using the NuSMV model checker for test generation from statecharts". In: *2009 15th IEEE Pacific Rim International Symposium on Dependable Computing, PRDC 2009* (2009), pp. 37–42. DOI: 10.1109/PRDC.2009.15.

[86] John Paul Kasse, Lai Xu, and Paul. T. de Vrieze. "The need for Compliance Verification in Collaborative Business Processes". 2018.

[87] John Paul Kasse, Lai Xu, and Paul de Vrieze. "A comparative survey of Business Process Verification Methods and Tools". In: *In Working Conference on Virtual Enterprises* (2017), pp. 355–367.

[88] John Paul Kasse, Lai Xu, and Paul de Vrieze. *PRO-VE camera-ready.* 2018.

[89] Mehreen Khan. *Companies face high cost to meet new EU data protection rules.* 2017.

[90] Moonzoo Kim et al. "Java-MaC: A Run-Time Assurance Approach for Java Programs". In: *Formal Methods in System Design* 24.2 (2004), pp. 129–155. ISSN: 09259856. DOI: 10.1023/B:FORM.0000017719.43755.7c.

[91] S. Kim and W. W Smari. "A Petri Net-based Workflow Modeling for a Human-centric Collaborative Commerce System". In: 5.Cd (2006), pp. 28–31.

[92] David Knuplesch et al. "On Enabling Data-Aware Compliance Checking of Business". In: (2010).

[93] Leslie Lamport. "Paxos Made Simple". In: *ACM SIGACT News* 32.4 (2001), pp. 18–25.

[94] K.G. Larsen, P. Pettersson, and Wang Yi Wang Yi. "Compositional and symbolic model-checking of real-time systems". In: *Proceedings 16th IEEE Real-Time Systems Symposium* (1995), pp. 76–87. ISSN: 1052-8725. DOI: 10.1109/REAL.1995.495198.

[95] Kim G Larsen, Paul Pettersson, and Wang Yi. "U PPAAL in a nutshell". In: (1997), pp. 134–152.

[96] Tingyu Liu, Yalong Cheng, and Zhonghua Ni. “Mining event logs to support workflow resource allocation”. In: *Knowledge-Based Systems* 35 (2012), pp. 320–331.

[97] Gavin Lowe. “Specification of communicating processes: Temporal logic versus refusals-based refinement”. In: *Formal Aspects of Computing* 20.3 (2008), pp. 277–294. ISSN: 09345043. DOI: 10.1007/s00165-007-0065-0.

[98] Linh Thao Ly et al. “SeaFlows Toolset Compliance Veri cation Made Easy 2 SeaFlows Toolset”. In: *German Research* (2007).

[99] Yongsheng Ma, Gang Chen, and Georg Thimm. “Change propagation algorithm in a unified feature modeling scheme”. In: *Computers in Industry* 59.2-3 (2008), pp. 110–118. ISSN: 01663615. DOI: 10.1016/j.compind.2007.06.006.

[100] Sara Mahdikhah et al. “A Business Process Modelling Approach to Improve OEM and Supplier Collaboration”. In: *Journal of Advanced Management Science* (2014), pp. 246–253. ISSN: 21680787. DOI: 10.12720/joams.2.3.246-253.

[101] Anu Maria. “Introduction to modeling and simulation”. In: *Winter simulation conference*. Vol. 29. 1. 1997, pp. 7–13.

[102] Jorge Tiago Martins et al. “Grounded theory in practice: a discussion of cases in information systems research”. In: *Information Systems Research and Exploring Social Artifacts: Approaches and Methodologies*. IGI Global, 2013, pp. 142–160.

[103] Rangarirai Matavire and Irwin Brown. “Investigating the use of grounded theory in information systems research”. In: *Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology*. ACM. 2008, pp. 139–147.

[104] SR McIntyre. *Integrated governance, risk and compliance: improve performance and enhance productivity in federal agencies*. Tech. rep. Technical report, PricewaterhouseCoopers, 2008.

[105] Alves de Medeiros et al. "Semantic process mining tools: Core building blocks". In: *ECIS* 2008 (2008), pp. 1953–1964. DOI: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.141.

[106] Patrick O Neil Meredith et al. "An overview of the MOP runtime verification framework". In: *International Journal on Software Tools for Technology Transfer* 14.3 (2012), pp. 249–289. ISSN: 14332779. DOI: 10.1007/s10009-011-0198-6.

[107] Marco Casassa Mont and Robert Thyne. "A Systemic Approach to Automate Privacy Policy Enforcement in Enterprises information lifecycle management A Systemic Approach to Automate Privacy Policy Enforcement in Enterprises". In: *Policy* (2006), pp. 118–134. URL: http://dblp.uni-trier.de/db/conf/pet/pet2006.html%7B%5C#%7DMontT06.

[108] Daniel L Moody. "The Method Evaluation Model : A Theoretical Model for Validating Information Systems Design Methods The Method Evaluation Model : A Theoretical Model for Validating Information Systems Design Methods". In: *ECIS* (2003).

[109] Shoichi Morimoto. "A Survey of Formal Verification for Business Process Modeling". In: *Computational Science – ICCS 2008 - Lecture Notes in Computer Science* 5102 (2008), pp. 514–522. ISSN: 03029743. DOI: 10.1007/978-3-540-69387-1_58. URL: http://www.springerlink.com/index/96x0051124530845.pdf.

[110] Jens Müller. "Security Mechanisms for Workflows in Service-Oriented Architectures". In: (2015).

[111] Tadeo Murata. *Petri Nets: Properties, Analysis and Applications.* 2008.

[112] Joyce Nakatumba. *Resource-aware business process management: analysis and support.* 2013. ISBN: 9789038634722. DOI: 10.6100/IR760115.

[113] Joyce Nakatumba and Wil MP van der Aalst. "Analyzing resource behavior using process mining". In: *International Conference on Business Process Management.* Springer. 2009, pp. 69–80.

[114] Kokash Natallia. “Integrating Compliance Management in Service-Driven Computing: Conceptual Models and Automation Architecture”. In: *Handbook of Research on Architectural Trends in Service-Driven Computing*. Centrum Wiskunde and Informatica (CWI), The Netherlands, 2014, p. 42.

[115] Thomas Neubauer, Markus Klemen, and Stefan Biffl. “Secure Business Process Management”. In: *Proceedings of the First International Conference on Availability, Reliability and Security* June (2006). DOI: 0-7695-2567-9/06. URL: http://portal.acm.org/citation.cfm?id=1130897.1130959%7B%5C&%7Dcoll=DL%7B%5C&%7Ddl=GUIDE%7B%5C&%7DCFID=111888922%7B%5C&%7DCFTOKEN=71872334.

[116] K Peffers et al. “A Design Science Research Methodology for Information Systems Research”. In: *Journal of Management Information Systems* 24.3 (2007), pp. 45–77. ISSN: 07421222. DOI: 10.2753/MIS0742-1222240302.

[117] Maja Pesic. *Constraint-based workflow management systems: shifting control to users*. 2008, p. 299. ISBN: 9789038613192. DOI: Urn:nbn:nl:ui:25-638413. URL: http://www.narcis.nl/publication/RecordID/oai:library.tue.nl:638413.

[118] Maja Pesic and Wil M. P. van der Aalst. “A Declarative Approach for Flexible Business Processes Management”. In: *Business Process Management Workshops* (2006), pp. 169–180. ISSN: 03029743. DOI: 10.1007/11837862_18.

[119] Maja Pesic, Helen Schonenberg, and Wil M.P. Van Der Aalst. “DECLARE: Full support for loosely-structured processes”. In: *Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOC* (2007), pp. 287–298. ISSN: 15417719. DOI: 10.1109/EDOC.2007.4384001.

[120] Carl Adam Petri. *General Net Theory*. 1977. URL: papers3://publication/uuid/80FAA443-1FAC-47FD-9828-B7D271194C80.

[121] Ping Jiang et al. “Interoperability of Cross-organizational Workflows based on Process-view for Collaborative Product Development”. In: *Concurrent Engineering* 16.1 (2008), pp. 73–87. ISSN: 1063-293X. DOI: 10.1177/1063293X07084640. URL: http://cer.sagepub.com/cgi/doi/10.1177/1063293X07084640.

[122] T. Elham Ramezani et al. “Compliance Checking of Data-Aware and Resource-Aware Compliance Requirements”. In: *On the Move to Meaningful Internet Systems: OTM 2014 Conferences. OTM 2014. Lecture Notes in Computer Science, vol 8841* 2 (2014), pp. 237–257. ISSN: 16113349 03029743. DOI: 10.1007/978-3-662-45563-0_14. URL: http://link.springer.com/10.1007/978-3-662-45563-0%7B%5C_%7D14.

[123] Marco Robol, Mattia Salnitri, and Paolo Giorgini. “Toward GDPR-compliant socio-technical systems: Modeling language and reasoning framework”. In: *Lecture Notes in Business Information Processing* 305 (2017), pp. 236–250. ISSN: 18651348. DOI: 10.1007/978-3-319-70241-4_16.

[124] Grigore Roşu, Feng Chen, and Thomas Ball. “Synthesizing monitors for safety properties: This time with calls and returns”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 5289 LNCS (2008), pp. 51–68. ISSN: 03029743. DOI: 10.1007/978-3-540-89247-2-4.

[125] Grigore Rosu and Klaus Havelund. “Synthesizing dynamic programming algorithms from linear temporal logic formulae”. In: (2001).

[126] A. Rozinat and W. M P van der Aalst. “Conformance checking of processes based on monitoring real behavior”. In: *Information Systems* 33.1 (2008), pp. 64–95. ISSN: 03064379. DOI: 10.1016/j.is.2007.07.001.

[127] Nick Russell et al. “Workflow data patterns”. In: *Business* 66.FIT–TR–2004–01 (2004), pp. –2004–01. DOI: 10.1007/11568322_23. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.80.6634%7B%5C&%7Drep=rep1%7B%5C&%7Dtype=pdf.

[128] Nick Russell et al. “Workflow data patterns: Identification, representation and tool support”. In: *International Conference on Conceptual Modeling*. Springer. 2005, pp. 353–368.

[129] SHAZIA W. SADIQ, OLIVERA MARJANOVIC, and MARIA E. ORLOWSKA. “Managing Change and Time in Dynamic Workflow Processes”. In: *International Jour-*

nal of Cooperative Information Systems 09.01n02 (2002), pp. 93–116. ISSN: 0218-8430. DOI: 10.1142/s0218843000000077.

[130] Shazia Sadiq and Guido Governatori. "Managing Regulatory Compliance in Business Processes". In: *Handbook on Business Process Management 2* 2008 (2010), pp. 159–175. DOI: 10.1007/978-3-642-01982-1_8.

[131] Shazia Sadiq, Guido Governatori, and Kioumars Namiri. "Modeling Control Objectives for Business Process Compliance". In: *5th International Conference, BPM 2007, Brisbane, Australia, September 24-28, 2007.* (2007), pp. 149–164. ISSN: 03029743. DOI: 10.1007/978-3-540-75183-0_12.

[132] Mattia Salnitri, Fabiano Dalpiaz, and Paolo Giorgini. "Modeling and verifying security policies in business processes". In: *Lecture Notes in Business Information Processing*. Vol. 175 LNBIP. 2014, pp. 200–214. ISBN: 9783662437445. DOI: 10.1007/978-3-662-43745-2.

[133] James Sanders and Dan Patterson. *Facebook data privacy scandal: A cheat sheet.* 2019. URL: https://www.techrepublic.com/article/facebook-data-privacy-scandal-a-cheat-sheet/ (visited on 04/18/2019).

[134] Ravi Sandhu. "Rationale for the RBAC96 family of access control models". In: *Proceedings of the first ACM Workshop on Role-based access control - RBAC '95* 1 (1996), 9–es. DOI: 10.1145/270152.270167. URL: http://portal.acm.org/citation.cfm?doid=270152.270167.

[135] D.R. dos Santos and S. Ranise. "On run-time enforcement of authorization constraints in security-sensitive Workflows". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 10469 LNCS (2017). ISSN: 16113349. DOI: 10.1007/978-3-319-66197-1_13.

[136] Adam Satariano. *Google Is Fined $57 Million Under Europe's Data Privacy Law.* 2019.

[137] S Shridevi and G Raju. "Ontology Based Annotation Verification Framework for". In: *Journal of Engineering and Applied Sciences* 13.9 (2018), pp. 2791–2798.

[138] Oleg Sokolsky. “Introduction to the Special Issue on Runtime Verification”. In: 41.December (2010), pp. 4–7.

[139] SOX. “Sarbanes-Oxley Act”. In: *USA Congress* (2002). URL: http://www.soxlaw.com/.

[140] Nair Srijith. *XACML Reference Architecture.* 2013. URL: https://www.axiomatics.com/blog/xacml-reference-architecture/ (visited on).

[141] Anselm Strauss and Juliet Corbin. *Basics of Qualitative Research.* 1990. DOI: 10.4135/9781452230153. arXiv: arXiv:1011.1669v3.

[142] Elham Ramezani Taghiabadi. *Understanding Non-compliance.* 2017. ISBN: 9789402804874.

[143] Kaijun Tan, Jason Crampton, and Carl A. Gunter. “The Consistency of Task-Based Authorization Constraints in Workflow Systems”. In: *Proceedings of the 17th IEEE Computer Security Foundations Workshop* (2004), pp. 155–169. ISSN: 1063-6900. DOI: 10.1109/CSFW.2004.1310739.

[144] Ahmed Tealeb, Ahmed Awad, and Galal Galal-Edeen. “Context-based variant generation of business process models”. In: *Enterprise, Business-Process and Information Systems Modeling.* Springer, 2014, pp. 363–377.

[145] Arthur H M Ter Hofstede et al. “Modern business process automation: YAWL and its support environment”. In: *Modern Business Process Automation: YAWL and its Support Environment* (2010), pp. 1–676. ISSN: 19454589. DOI: 10.1007/978-3-642-03121-2. arXiv: arXiv:1011.1669v3.

[146] H Thomas and E James. “The New Industrial Engineering : Information Technology And Business Process R ...” In: (1990).

[147] R.K. Thomas and R.S. Sandhu. “Conceptual foundations for a model of task-based authorizations”. In: *Proceedings The Computer Security Foundations Workshop VII* May (1994), pp. 66–79. ISSN: 1063-6900. DOI: 10.1109/CSFW.1994.315946. URL: http://ieeexplore.ieee.org/document/315946/.

[148] Roshan K. Thomas and Ravi S. Sandhu. "Task-based Authorization Controls (TBAC): A Family of Models for Active and Enterprise-oriented Authorization Management". In: *Database Security* 11 (1997), pp. 166–181. DOI: 10.1007/978-0-387-35285-5_10. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.54.6227%7B%5C&%7Drep=rep1%7B%5C&%7Dtype=pdf.

[149] Wil M P Van Der Aalst. "Woflan: A Petri-net-based Workflow Analyzer". In: *Systems Analysis Modelling Simulation* 35.3 (1999), pp. 345–357. ISSN: 0232-9298. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.80.6992%7B%5C&%7Drep=rep1%7B%5C&%7Dtype=pdf.

[150] Wil MP Van Der Aalst. "Business process simulation survival guide". In: *Handbook on Business Process Management 1*. Springer, 2015, pp. 337–370.

[151] Wil MP Van der Aalst et al. "Business Process Simulation: How to get it right". In: *BPM Center Report BPM-08-07, BPMcenter. org* 285 (2008), pp. 286–291.

[152] H. M W Verbeek, T. Basten, and W. M P Van Der Aalst. "Diagnosing workflow processes using Woflan". In: *Computer Journal* 44.4 (2001), pp. 246–279. ISSN: 00104620. DOI: 10.1093/comjnl/44.4.246.

[153] H.M.W. Verbeek, Wil M.P. van der Aalst, and A.H.M. ter Hofstede. "Verifying Workflows with Cancellation Regions and OR-joins : An Approach Based on Relaxed Soundness and Invariants". In: October 2017 (2007). DOI: 10.1093/comjnl/bxl074.

[154] H.M.W. Verbeek, W.M.P. van der Aalst, and Akhil Kumar. "XRL/Woflan: Verification and Extensibility of an XML/Petri-Net-Based Language for Inter-Organizational Workflows". In: *Information Technology and Management* 5.1/2 (2004), pp. 65–110. ISSN: 1385-951X. DOI: 10.1023/B:ITEM.0000008077.91413.86. URL: http://link.springer.com/10.1023/B:ITEM.0000008077.91413.86.

[155] Atluri Vijay. "Security for workflow systems". In: *Information Security Technical Report* 6.2 (2001), pp. 59–68. ISSN: 9780387485331; 0387485333. DOI: 10.1007/978-0-387-48533-1.

[156] Janice Warner and Vijayalakshmi Atluri. "Inter-instance authorization constraints for secure workflow management". In: (2006), p. 190. DOI: 10.1145/1133058.1133085. URL: http://portal.acm.org/citation.cfm?id=1133085.

[157] Janice Warner et al. "Using Semantics for Automatic Enforcement of Access Control Policies among Dynamic Coalitions". In: *SACMAT '07: Proceedings of the ACM symposium on Access control models and technologies* (2007), pp. 235–244. ISSN: 03029743. DOI: 10.1145/1266840.1266877.

[158] Hillel Wayne. *Learn TLA+*. 2018. URL: https://learntla.com/introduction/ (visited on 01/17/2019).

[159] Mathias Weske. *Business ProcessManagement Concepts, Languages, Architectures.* 2007. ISBN: 9783540735212.

[160] Stephen White. "Introduction to BPMN". In: *IBM Cooperation* 2.0 (2004), pp. 1–11. ISSN: 09636897. DOI: 10.3727/000000006783982421.

[161] W.M.P. van der Aalst. "Workflow Verification: Finding Control-Flow Errors Using Petri-Net-Based Techniques". In: *Business Process Management* 1806 (2000), pp. 19–128. DOI: 10.1007/3-540-45594-9_11.

[162] Peter Y.H. Wong and Jeremy Gibbons. "Formalisations and applications of BPMN". In: *Science of Computer Programming* 76.8 (2011), pp. 633–650. ISSN: 01676423. DOI: 10.1016/j.scico.2009.09.010.

[163] M T Wynn et al. "Verifying Workflows with Cancellation Regions and OR-joins : An Approach Based on Reset Nets and Reachability Analysis". In: *Business Process Management* (2006), pp. 389–394.

[164] M.T. Wynn et al. "Business process verification – finally a reality!" In: *Business Process Management Journal* 15.1 (2009), pp. 74–92. ISSN: 1463-7154. DOI: 10.1108/14637150910931479. URL: http://www.emeraldinsight.com/doi/abs/10.1108/14637150910931479.

[165] Lai Xu. "Monitorable Electronic Contract." In: *CEC*. 2003, p. 92.

[166] Lai Xu. "MONITORING MULTI-PARTY CONTRACTS FOR E-BUSINESS Lai Xu". PhD thesis. 2004. ISBN: 9056681281.

[167] Lai Xu and Manfred A Jeusfeld. "Pro-active monitoring of electronic contracts". In: *International Conference on Advanced Information Systems Engineering*. Springer. 2003, pp. 584–600.

[168] Lai Xu, Manfred A Jeusfeld, and Paul WPJ Grefen. "Detection tests for identifying violators of multi-party contracts". In: *ACM SIGecom Exchanges* 5.3 (2005), pp. 19–28.

[169] Sergio Yovine. "Kronos: A verification tool for real-time systems". In: *International Journal on Software Tools fYovine, S., 1997. Kronos: A verification tool for real-time systems. International Journal on Software Tools for Technology Transfer, 1 (1–2), 123–133.or Technology Transfer* 1.1-2 (1997), pp. 123–133. ISSN: 14332779. DOI: 10.1007/s100090050009.

[170] E. Yuan and J. Tong. "Attributed Based Access Control (ABAC) for Web Services". In: *The IEEE International Conference on Web Services*. 2005, pp. 561–569.

[171] Jörg Ziemann and Thomas Matheis. "Modelling of cross-organizational business processes-current methods and standards". In: *Proc. EMISA '07* 2.2 (2007), pp. 87–100. DOI: http://dx.doi.org/10.18417/issn.1866-3621. URL: http://doc.utwente.nl/64399/1/EMISA-Proceedings.pdf%7B%5C#%7Dpage=87.

[172] Xiangrong Zu. "A Role and Task-based Workflow Dynamic Authorization Modeling and Enforcement Mechanism". In: (2009), pp. 1593–1596.