QUERY DATA INCONSISTENCY FOR BUSINESS PROCESSES

By

Yongping TANG

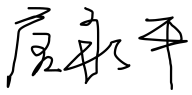
A THESIS SUBMITTED TO MACQUARIE UNIVERSITY
For The Degree Of Master Of Research (MRes)
Department of COMP
December 2018

MACQUARIE
University
SYDNEY·AUSTRALIA

Typeset in LaTeX 2$_\varepsilon$.

Declaration

I hereby state that this thesis contains only my own original work except where explicit references have been made to the work of others, and has not been submitted for any other degree to any other university or institution.

(Signed)  Date: 8 Nov 2018

Yongping TANG

.

Acknowledgements

First and foremost, I want to thank my supervisor Professor Jian Yang, who has brought me to the acdamic world of BPM, accompanied me to overcome the difficulties along the way, and encouraged me to pursue excellence. I would also like to thank Prof Su and Dr Williams Zhao, who helped me a lot during the journey in the Master of Research project and thesis.

Whenever I am overwhelmed or need help, Professor Yang and Dr Zhao are always there to show me the correct direction and provide me necessary instructions. I have no doubt that the skills I have learned from them will stay with me throughout my life and help to maximize the outcome of researches. My gratitude towards them is beyond words.

During my MRes degree, the excellent courses in how to conduct research provided by computing department of Macquarie University have shaped my knowledge base and enriched my skill sets. I definitely appreciate the lectures and trainings provided by Macquarie University.

Last but not least, I am grateful to my family. My wife Qian Zhou and daughter Zhouwunie Tang. I am not able to achieve what I have achieved without their whole-hearted support and encouragement. They are the people who care me; who give me strength to take on any challenges that are ahead of me; and who always stand by me and make everything in my life simpler.

This thesis is dedicated to my wife and daughter who support me to pursuit my personal dream.

Abstract

Business processes are designed to achieve business goals under procedural rules by orchestrating tasks, information and documents. Managing data inconsistency in business processes is a challenging task. If not managed properly, business will face negative financial consequences. For instance, in a typical service business process, dispatching invoice depends on service being ready. When starting to dispatch invoice, if the service is not yet deployed, the invoice will be wrong and there may be financial damage to the business. From literatures, some work provides modelling for business process execution with notations of activities, events and flows and deals with inconsistency problem by patterns; some work analyses data generated in business process and investigates the reachability between data points. Although substantial works have been done, the data inconsistency problem has not been properly resolved. In particular, it is still lacking of modelling language and resolution for inconsistency caused by multiple starting points of business processes and dynamics of business processes execution.

This thesis provides data consistency solution in three aspects: a business process modelling in enriched business workflow notation with data states and temporal properties, a classification of data consistency categories, and a workflow query algorithm to discovery data inconsistency issue.

Keywords: business process, provenance, workflow specification, reachability, FSM, DAG, BPMN, temporal constraints, inconsistency

Contents

1

Introduction

In this chapter, background on Business Process Model and Notation and Data Provenance will be first introduced. Two additional business process modelling of Petri net and FSM will also be introduced, followed by some conceptual models including YAWL, UML and temporal properties.

1.1 Background and BPM Workflow

A business process is a set of logically related activities that, when performed in the appropriate sequence, and according to the correct business rules, meets a positive business objective. ([1]) Business process management is to support business process using methods, techniques and software to design, enact, control and analyse operation process including human, organisations, applications, documents and other sources of information. ([2]) Therefore, technology of business process management is to a few extent related to the workflow technology which means passing document or information from person to person (or from system to system) in enterprise presented as flowchart alike diagrams. When the document or information in incorrectly transferred or not transferred at right time, data inconsistency in business process occurs. The data inconsistency

has negative impact on business objective. In the rest of thesis, business process management is referred as BPM and the workflow technology for business process is referred as BPM Workflow or business workflow.

There are few key elements to be considered in BPM workflows which provide more context on data inconsistency problem:

1. tasks, the atomic elements involved in any BPM workflow
2. roles or actors, the human being or machine system who execute the tasks
3. objects or data, the artefacts which are manipulated by tasks
4. links, the associations between processor tasks and successor tasks in BPM workflow execution sequence

From literatures, there are two major research approaches looking into this issue, some are from business workflow execution perspective, and others are from business workflow data perspective.

1.1.1 Business Process Model and Notation

BPMN approach focuses on design time workflow pattern and process sequence to find logical indications on the problem. The most important business process modelling approach is called Business Process Model and Notation (BPMN) ([3]). BPMN defines sets of graphical elements to represent business workflows. There are three major categories of elements: flow objects, artefacts and connection objects.

The flow objects include ***activities*** which represent tasks conducted inside certain business process step, ***events*** which happens randomly or by schedule, ***gateways*** which represent decision choices in different business process execution paths. Artefacts include ***data object***, ***group*** of sets of elements, and ***annotation*** to elements. Connecting objects are representing either sequential execution with ***sequence flow*** arrow, message sender and receiver with ***message flow***, or ***relationship*** between elements by associations. BPMN will be detailed in section 2.2.

1.1.2 Data Provenance

Data provenance approach looks into the history of all data transitions and find out missing business workflow footprint patterns, which are called workflow data provenance, and targets to resolve the problem through analysis on the data relationships and dependencies. This approach typically defines activity in workflow as module. Atomic module includes one activity only. Composite module includes multiple activities, other atomic or composite modules. Inputs and outputs of each module are specified as data points. Mapping between inputs and outputs are specified internally in atomic module. The execution of a workflow is typically modelled as workflow production. Workflow production generates an execution path in graph representation which is constructed with only atomic module. Data points are labelled according to workflow module definition and sequence.

Using the workflow data provenance model, data point reachability can be analysed through labelling of the data point and module. The reachability analysis examines whether a particular input or output data point impacted by another input or output data point. For complex business workflow, specific rules called workflow $composition$, $substitution$ and $propagation$ are defined to merge, replace, transfer data points between workflow modules respectively. Details on workflow provenance can be found in section 2.2.

1.1.3 Other Business Process Models

In addition to BPMN, there are two other major models of business processes.

Petri Net

Petri net is introduced in Carl Adam Petri's Ph.D. thesis ([4]). Petri net provides the following key elements: 1) places, graphically represented by circles and physically represents input and output data points in business processes; 2) transactions, graphically represented by rectangles and physically represents activities in business processes; 3) direct arcs which connect places and transactions, represents relationship between data and activities; 4) token, capture run time behaviour and is annotated as an overlay on Petri net diagram.

Based on original Petri net, there are also enhancement such as coloured Petri net where annotations are associated with places, transactions and arcs. Details will be explored in chapter 2.

FSM

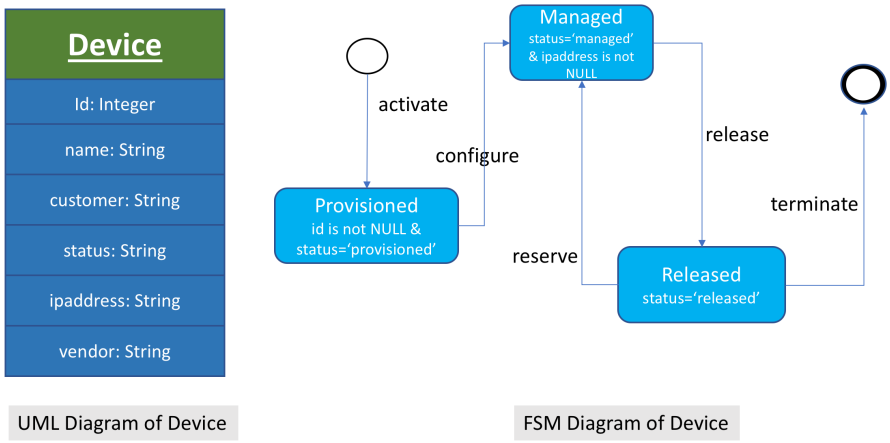


FIGURE 1.1: Sample UML and FSM Diagram

Finite State Machine (FSM, [5]) is another modelling used to specify the transition of data object states. The data objects are be associated with activities in business processes. FSM defines state transition on a data object. As shown in the sample FSM on the right in Figure 1.1, there are three states in addition to initial state and termination state. Activities are noted in each state transit. In the example, *configure* is the activity triggering state transition from *Provisioned* to *Managed*. In additional, each state contains some specific conditions on attributes of the data object. For instance, in *Released* state, the condition is status should be equal to *released*.

1.1.4 YAWL, UML and Temporal Properties

YAWL stands for Yet Another Workflow Language which provides additional control logic comparing with BPMN. YAWL provides the semantics of condition, multiple instances, split and join task on business process workflow. The major benefit of YAWL is that it provides descriptive information for multiple instances, data synchronisation and global data cascade behaviour. Details can be found in subsection 2.1.

UML stands for Unified Modelling Language ([6]). UML class diagram provides conceptual details on data objects including attributes, operations and relationships between objects. In the rest of thesis, UML class diagram is referred as UML diagram for simplification. A sample UML diagram can be found on the left in Figure 1.1. The diagram describes a device object class which includes six attributes: id, name, customer, status, ipaddress and vendor.

Timing is anther conceptual aspect to be considered in business workflow. Any business goal only makes sense when taking timelines into consideration. Each activity may have a deadline and duration as properties, and may also happen only after or before another activities. These becomes temporal constraints to be specified on top of BPMN. One type of temporal constraint is to specify the logical consequence between activity, event and state. For example, a function $wasAction$ to find out which activity leads to a specific state. Anther type of temporal constraints is temporal logic operators where order sequence can be specified between activities and states. For example, one state must be entered before execution of one named activity.

This thesis describes a research problem in the direction of BPMN with enrichment. In particular, solution on data inconsistency problem is investigated. A query algorithm is developed based on combined specification of BPMN, UML, FSM and temporal properties. The research problem and the motivation scenario is illustrated in next section.

1.2 Motivation Scenario and Problem Statement

An example in telecommunication service fulfilment and assurance is investigated. In this example, multiple business processes are collaborated by multiple business partners - ***Customer***, ***Customer care representative***, ***OSS engineer***, ***Field engineer***. The meaning of the participants are explained in section 1.2.1. These participants interact with different systems shown in Figure 1.2.

1.2.1 Business Participants and Sub Business Processes

Figure 1.2 lists major systems in a telecommunication service provider. The systems manage customer new order and provide quality assurance on the device and service. A set of workflows are designed to meet the overall business goal: provide network service to customer in time, and ensure customer is satisfied on the service. In this example, the following participants are interacting with different subprocess including Order Management (OM), Provisioning (PROV), Fault Management (FM) and Inventory.

- ***Customer*** *Customer* orders a new Internet connection service from telecommunication company, by interacting with *OM* to submit the order. *Customer* may also interact with *Customer Care* system to raise complaints on the service. The service typically runs from a network device called Customer Premises Equipment or CPE.

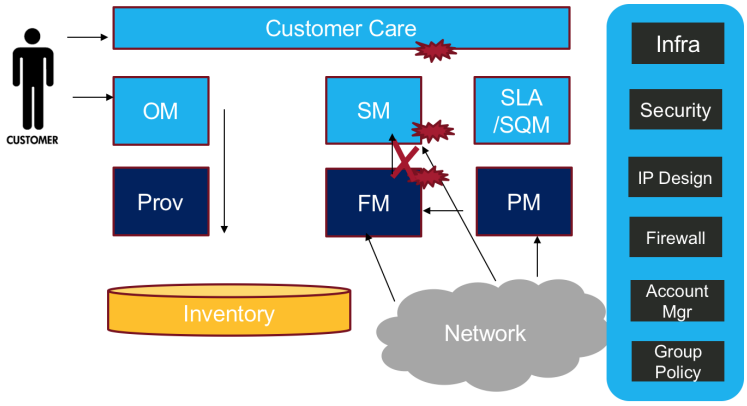


FIGURE 1.2: Major systems managing customer order and service assurance

- ***Customer care representative*** *Customer care representative* operates *Customer Care* system to receive customer's service complaint, conduct troubleshooting on customer's service and also interact with SM system to raise trouble tickets in order to restore service.
- ***OSS engineer*** *OSS* stands for Operation and Support Systems which are referring to the network management systems. *OSS engineer* manages telecommunication service provider's network, by interacting with $Prov$, FM and PM to performance resource provisioning ($Prov$), fault management (FM) and resource performance management (PM).
- ***Field engineer*** *Field engineer* interacts with SM system which fixes device issues if service problem is caused by the network device. *Field engineer* typically visits *Customer*'s premises to fix device problems.

1.2.2 The Business Process Scenario

In telecommunication business process, a new service order will be first fulfilled by OM and $Prov$ so that customer can get a device and is able to connect to Internet on first place. The device is activated during the order fulfilment process and device information is then stored in $Inventory$ system so other systems can get details of this device when required. FM system is functioning to monitor the fault status of the customer device. In the event of device failure, there will be minimum one event sent to FM. FM then finds out which device the events are associated with by comparing the information in the event and the device information in $Inventory$. Once device information is associated with event, FM provides faulty event status to customer care team who

can in turn dispatch work order to fix the device. Data inconsistency issue occurs in particular when *FM* is not able to identify which device the event should be associated to or a wrong device is identified due to the inconsistency of device information. When the issue happens, the company either has no visibility on the faulty device and no proactive action taken in resolving customer's device, or dispatches a job on a wrong device location and encounters substantial financial loss.

Two business process scenarios are described in Figure 1.3 and 1.4.

Figure 1.3 shows a good scenario where *Customer* gets support and feedback from *Customer Care* in acceptable timeframe. The process starts when *Customer* places an order for Internet connection service. *OSS Engineer* needs to ensure device (CPE) is ready to serve the *Customer* through Fulfilment function. Here the Fulfilment includes both *OM* and *PROV* described earlier. In the following description, sequence number of the activity will be added as #) for easy reference. The CPE information will be updated into *inventory* system by *3): update inv*. The inventory also provides necessary configuration on the CPE to ensure it is well configured for both external and internal systems. At same time, the CPE will also return Fulfilment system that it is *4): ready to use* and in turn *Fulfilment* informs *Customer 5): order fulfilled*. The *Assurance* functions include both *FM* and *PM*. The two assurance functions get all device information from *inventory* by *9): sync CPE info* and start to monitor the fault and performance status. At a point of time, there will be updates from CPE by any *11): fault & performance issue* and the *Assurance* will inform *Customer Care* by *12): report CPE issue*. When the device issue impacts *Customer*, textitCustomer will raise *13): complain* to *Customer Care*. As the *Customer Care* is already informed about the issue, it will *14): get details* from *Assurance* (*FM* and *PM*) and dispatch job to *16): resolve issue* and provides *Customer* explanation and ensure that the service is in good status.

Figure 1.4 shows a bad scenario where *Customer* is not able to obtain reasonable feedback from *Customer Care* timely because the information is not updated in systems in time. For example, when *Assurance* tries to *7): sync CPE info* from *inventory*, it only retrieves partial information due to *13): policy compliance* activity in *CPE* is delayed and *inventory* has not got confirmation of *16: configured*. As the order is still fulfilled and *Customer* has started to use the service, there is a chance that the CPE facing issue and reports *9): fault & performance issue* to *Assurance*. Due to the lake of CPE full information, *Assurance* is not able to analyse the issue and not able to report the issue to *Customer Care* at this point of time. When *Customer 11): complains* the issue, *Customer Care* gets *14): empty return* from *12): get details* and is *15): not able to resolve* the issue. *Customer Care* is then providing *17): NO explanation* to *Customer*. The *Customer* will not be happy on not getting any reasonable explanation from *Customer Care* and may decide to terminate the

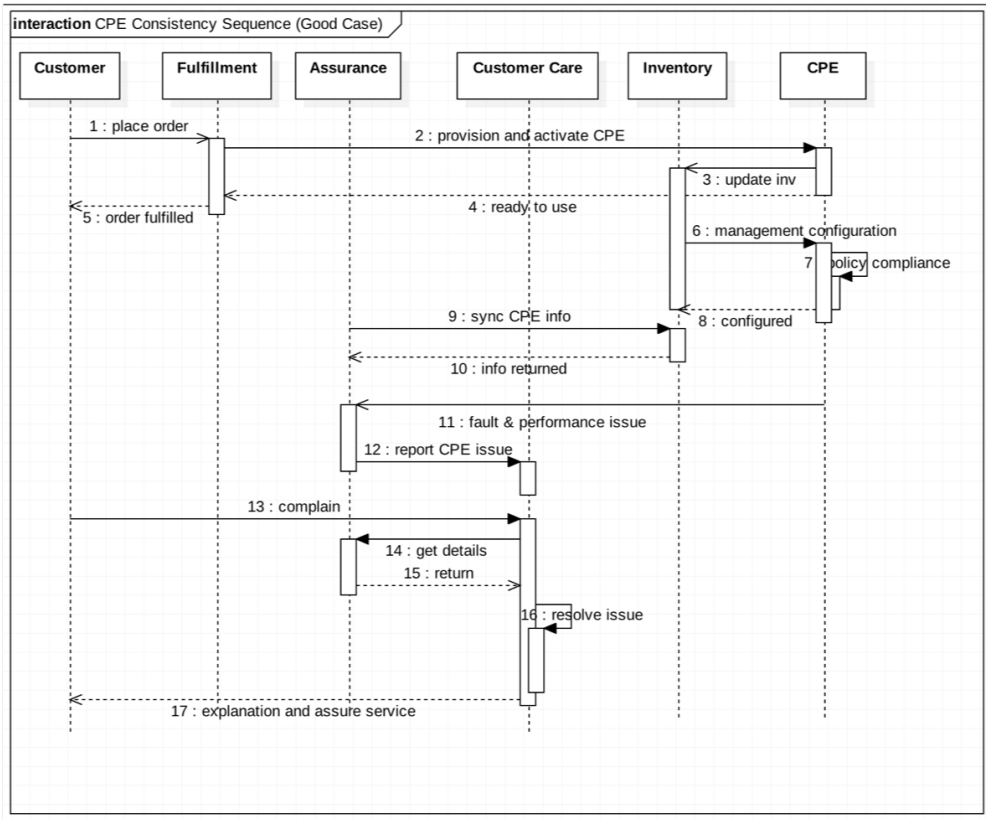


FIGURE 1.3: Sequence Diagram of Good Scenario

service contract with this telecommunication service provider and switch to another provider. This potentially will cause financial loss to the telecommunication service provider company. Although, CPE informs *16): configured* to inventory at step 16 eventually, *Assurance* is able to get full details and *Customer Care* is able to *22): resolve* the issue and *23): inform customer*, it may be too late to change *Customer* mind.

In this business process, in good scenario, data is updated correctly between different participants and sub business processes in a consistent way. In contrast, data is not consistent in bad scenario where the activities in CPE on *policy compliance* is delayed which blocks *Assurance* and *Customer Care* from getting correct device inventory and fault & performance information. This is a typical data inconsistency issue. A problem statement will be provided in next subsection.

The data inconsistency issue is not a trivial problem and requires systematical approaches in identification and resolution. In particular, an algorithm to identify potential data inconsistency risk without actually executing the workflow instances is beneficial to businesses technically and financially.

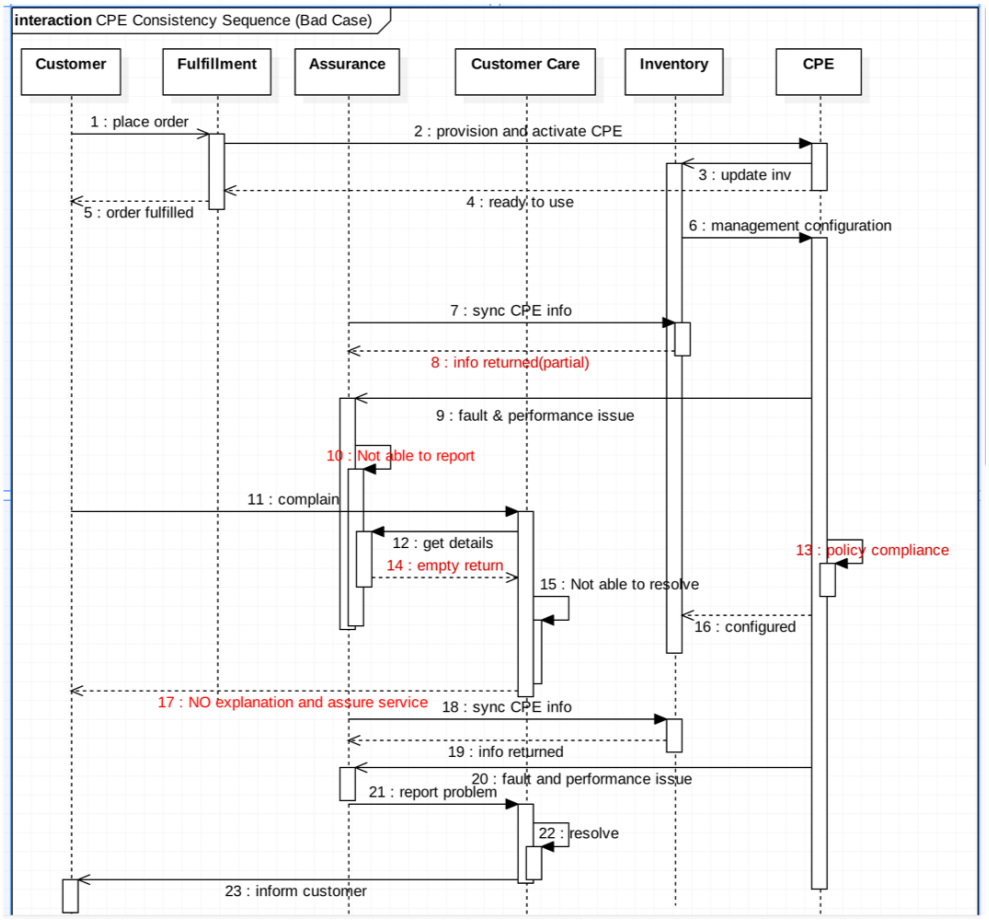


FIGURE 1.4: Sequence Diagram of Bad Scenario

1.2.3 Problem Statement

To identify and resolve the issue, it is required for a proper query in the business process to find out dependencies between workflow modules and identify potential data inconsistency issue. Reachability between input data and output data is one possible criteria to be examined and related work will be introduced in next chapter. Overview of existing work is first explored in next section.

The thesis focuses on the following three problems:

- how to model business process and underpin the data and temporal properties
- what are the typical patterns of data inconsistency scenarios
- how to identify potential data inconsistency issue in a given business process design

1.3 Existing Work Overview

A major trend is identified on how the research in business workflow (including scientific workflow) evolves and especially on how data transition and migration in workflow is studied. A major difference in scientific workflow is that it involves large amount of structured data and deterministic execution path comparing with business workflow. The research trend demonstrates the steady progress to address the data inconsistency issue. The chronological highlights are listed below:

- introduction of general business process model ([7])
- study on how data object evolves and migrates were studies in Su [8]
- model on business processes, introducing Business Process Model and Notation (BPMN) ([3]) and PetriNet ([3])
- theory on how data should be annotated and transferred were set up by researchers such as Tova Milo ([9]), Davidson Susan ([10]), Freire Juliana ([11]) and Val Tannen ([12]). These were done in the years from 1998 to 2011
- study on workflow model and resolve particular questions such as data reachability is examined in the work from Bao ([13]) under the guidance of Milo and Susan, and some other related works. These were in the years from 2012 to 2014
- continuous evolvement in research on data provenance, model and the applications.
- evolution on BPMN with Decision Model Notation ([14])

In [8] Su studies extensively on the dynamic constrains on data object migration between applications, although the work looks at dynamic aspect of data migration, business workflow activity and execution model are not considered extensively; [9] provides an data provenance implementation approaches based on Hadoop Pig, details on modelling provenance data, however, data dependencies between different executions are not addressed; [13] provides an effective algorithm to determine reachability between data nodes, however, the algorithm addresses scientific workflow only and does not take into consideration the complex data model and dynamic execution characteristics in business workflow; [14] extends BPMN with Decision Model Notation and provided a mechanism to describe decision nodes with more data context information, however this technique is yet to be applied to data inconsistency issue. The concepts and general methodologies on the related work are introduced in chapter 2.

1.4 Contribution of the Thesis

This thesis provides data inconsistency solution in the following three aspects:

Introduction of Time and Data enriched BPMN (TD-BPMN)

TD-BPMN describes a set of business processes using three inter-related diagrams, including BPMN, UML Class Diagram and Finite State Machine. TD-BPMN provides insights on data model view and how data states are transited over time.

Categories of data inconsistency patterns

Classification of data inconsistency is identified in this thesis with examples. The patterns implied in the different categories differ according to various combination of activity, data and time.

Query algorithm to find data inconsistency for business processes

This work provides a data inconsistency identification process based on workflow specification defined in TD-BPMN. Algorithm efficiency and performance results are captured and analysed.

1.5 Thesis Organisation

The rest of this thesis is organised as following: In chapter 2, common background knowledge on two areas of related work are briefly introduced. This includes two major methodologies: 1) workflow model and notation methodologies are introduced and 2) data provenance and workflow provenance are generally reviewed. Basic modelling of the research problem in BPMN is also explored. Related studies on workflow query are also introduced in this chapter. In particular data provenance labelling and reachability query is explored in details. The method resolved partially the data inconsistency problem. The limitations on related work are also discussed and the gaps are identified. In chapter 3, three major contributions of the thesis are provided. This includes TD-BPMN, data inconsistency categories classification, and the methodology of identifying potential inconsistency in a given workflow. In chapter 4, results of the data inconsistency query implementation is presented and finding from the results is analysed with limitations explained. In chapter 5, as a conclusion of this thesis, the contributions are summarised together with ideas on future work.

2

Literature Review

In this chapter, the following related work will be explored: BPMN Model, Data Provenance, and Workflow Patterns and Queries.

2.1 BPMN Model

BPMN and Related Work

A comprehensive description of business process management are provided in the BPM book from Weske [3]. Typical descriptive languages and methodologies in modelling a business process are defined and used by engineers and researchers across the globe, including PeriNet([3]), BPMN([3]), YAWL([15]), FSM ([5]) and UML ([6]).

One standard business process modelling is Business Process Modelling Notation (BPMN). The intent of the BPMN is relatively similar to the intent of the Unified Modelling Language ([6]) for object-oriented design and analysis. It comes out by identifying the best practices of existing approaches and combining them into a new, generally accepted language. Figure 2.1 outlines

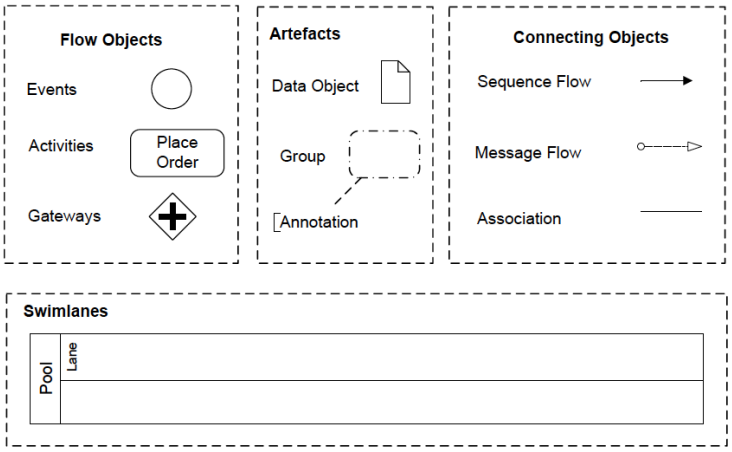


FIGURE 2.1: Business Process Modelling Notation (BPMN): categories of elements, from [3]

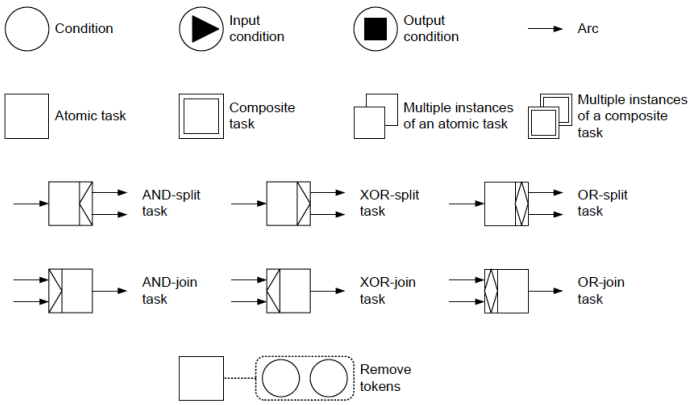


FIGURE 2.2: Notational elements of YAWL, van der Aalst and ter Hofstede (2005), from [3]

major elements in BPMN diagram.

Although different modelling methods have their advantages and disadvantages, BPMN is considerably as standard in both research and industrial projects.

BPMN evolves in areas of decision making model and event handling. A special BPMN was introduced as uBPMN-Based Patterns for modelling Ubiquitous Business Processes in [16]; optimal acquisition of input data for decision taking in business processes is discussed in [17]; implementation on handling events in business process implementation by early subscription and event buffering is studied in [18]; a framework for integrating real-world events and business processes in an IoT related environment is introduced in [19]; methods on integrating behavioural aspects

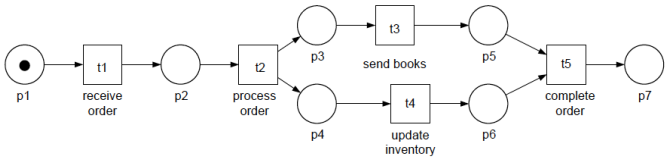


FIGURE 2.3: Sample Petri net representing single process instance, from [3]

and label analysis in order to match events and activities are explored in [20]. Last but not least, a Decision Model Notation with a data centric approach is formalised in a latest work in [14] by Prof M. Weske and the team. As explained in previous chapter, workflow condition check is impacted by the data consistency problem in particular, and logically a condition check is a decision making action. Also event driven workflow nodes are dynamic in term of timing, which is one of the challenges leading to the data inconsistency problem.

Petri net is introduced in Carl Adam Petri's Ph.D. thesis ([4]) which provides additional process information comparing with BPMN. Carl generalises automata theory by concurrency and introduces a new modelling approach. The approach provides not only a graphical representation but also an equivalent mathematical formalisation, which is called Petri net. In Petri nets, the static structure is represented by a Petri net, and the dynamic behaviour is captured by the token putting as an overlay layer on the Petri net. The key elements in Petri nets include places, transitions, and directed arcs which connect places and transitions. In graphical notations, places are represented by circles, transitions by rectangles, and connectors by directed arcs. Transitions have input and output places. Figure 2.3 shows a sample Petri net of a single process instance. To provide more informative details on the data in workflow node, a coloured Petri net is also introduced as shown in Figure 2.4 on page 15. In the sample, three employees data with their salaries are annotated to place $p1$; if the salary is more than 20,000, AccessRisk connects to $p2$, otherwise connects to $p3$. There are other extensions in Petri net model for different problems. For example, a time Petri net is introduced in [21] which looks into concurrency in business processes; a parallel algorithm for Petri object simulation in stochastic Petri net is developed in [22]; a wider Petri net called Open net is discussed in [23].

Under BPMN and Petri net, it is still lacking of a process language that directly supported all control flow patterns. The motivation for the development of Yet Another Workflow Language (YAWL, [15]) is to address the limitation. While YAWL uses workflow nets as a major ingredient, the execution semantics of process instances is specified by state transition systems and not by

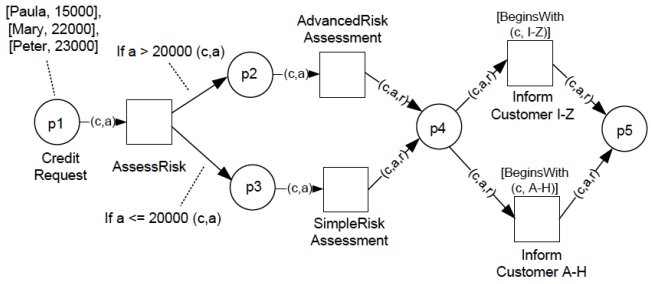


FIGURE 2.4: Sample coloured Petri net, from [3]

Petri nets. The key elements in YAWL are the control flow patterns and the execution semantics. Figure 2.2 presents all notational elements in YAWL. The YAWL notations provide richer control flow elements such as Condition, Input condition, Output condition, AND-split, XOR-split. The focus of YAWL is on activity transactions so one of the limitation is that data transitions details are not captured.

There are limitations on the workflow model based approach. In particular, it does not provide details on what are the exact states device data and Alarm, and what are the different conditions of these states. These details are required in inconsistency identification.

2.2 Data Provenance

Another category in workflow research is Data Provenance. It looks into the data generated and operated in workflow.

The data provenance provides a practical method in modelling and addressing data reachability issues. In workflow management, particularly on scientific workflow, there are mainly two types of workflow provenance management: workflow design time provenance and workflow run time provenance. Design-time provenance is focusing on provenance data coming from workflow meta model in the workflow specification, including inputs and outputs of each module in the workflow design. Run-time provenance focuses on the query of meta data, execution log and dependencies. The study on these two types of provenance management also includes developing a meaningful and functional scientific workflow management system (SWMS). The key difference between design-time provenance is: the run-time provenance requires more system resources in term of CPU, memory and storage as there could be workflow modules executing in parallel and some modules may even executed recursively.

Workflow data provenance approach looks into how to define workflow, how to specify workflow execution and how to examine and query workflow data relationship and reachability. The details on data provenance and related work are listed below.

Related Work in Workflow Provenance

Provenance is studied with the help of international forums being set up for people to development solutions for provenance problem challenges. In table 2.1, some provenance challenge problems are listed. The notions of Why, How, Where, Which, What-if provenance is mainly for different types of dependencies, and they are quite similar as the data trace concept in database management. These provenance types study both forward and backward provenance. Normally, Why, Where, How provenance types belong to backward provenance, looking for data model design, data source, and data generation respectively; Which provenance belongs to forward provenance, as it looks for predecessor of input data to a workflow; What-if provenance has the characteristics of both forward and backward provenance, and it studies the impact to forward or backward data flow by assuming a data change.

No.	description	intro time	type
1	list out possible SMF inputs and execution path which impacts data generation	First	How Provenance
2	list out possible output data impacted by certain inputs	First	Which Provenance
3	list out possible processes which are mandatory for generating particular data	Third	What if Provenance
4	list out possible data sources of partial data	Third	Where Provenance
5	list out possible inputs generated by some data	Third	Why Provenance

TABLE 2.1: few different queries in Provenance Challenge

representation of workflow Most scientific workflows utilise terms of Directed Acyclic Graph (DAG, a graph with nodes and directed edges, typically a workflow run can be represented using DAG) as presentation method, and a minority of systems (such as GridFlow) utilises Petri net. Although Petri net is based on foundational mathematics theories, good modelling capability and expressive method, comparing with DAG, it is less self-expressive, and hard to be understood by normal reader due to complexity. In oder to enhance language express capability, in most of the scientific workflow management system specific semantics are defined. Some system, such as

BPEL Power ([24]), utilises Business Process Execution Language (BPEL) which is widely used in Business workflow ([25]). Pegasus ([26]) system provides a DAX language which has underlining data flow model. The limitation is that it does not provide control flow such as data iteration and branch. In contrast, BPEL which is based on control flow provides many control flow structures such as Fork, While, If and Switch. These differences in control flow are mainly caused by the data driven characteristics in scientific workflow, and it is much different from business workflow which is more logic driven and is lacking of workflow control. In order to resolve the lack of workflow control flow, Taverna ([27]) is introduced which provides a set of methods to control workflow. Although Taverna does not provide explicit parallel and iteration execution, it supports implicit parallel and iteration execution based on data set. Taverna also provides Control Links which associates two processes without a hard data dependencies, and it is able to force the launch of a process to be delayed until the other completed.

design-time provenance related systems A typical use case in design time provenance is version control, and some SWMS systems utilise file version control system to manage design time provenance. RAMP ([28] and Lipstick ([9]) are deployed in Hadoop platform. As the underlying HDFS is Append-Only, and it supports version control naturally. VisTrails ([11]) defines a set of version tree for scientific workflow, each node in the tree represents a certain version of the scientific workflow, edge in the tree represents the evolution from one version to another version. Version tree is Append-Only, so user is able to revert back to any previous version.

general model in provenance Data provenance model is core in provenance systems. Sometimes, for particular use case, some system provides proprietary provenance data model. However, due to the fact that more and more requirement arises in interoperation, more and more provenance systems select a general and open data provenance model. Two provenance models are widely used: OPM ([29]) and PROV ([30]). OPM was raised in third provenance challenge. It contains a design principle on data artefact and relationship which impacts also many other provenance model. OPM contains the following aspects: abstract Model, OPMX (XML based), OPMV (vocabulary), OPMO (OPM Object), and OPM4J (JAVA API). On the other hand, PROV model and the PROV object (PROV-O) are becoming W3C standard, and also generate the related working draft such as PROV-XML, PROV-AQ, PROV-DICTIONARY. Some specific provenance model system, such as (Kepler, [31]) provides options to convert internal model to OPM or PROV. From modelling capability point of view, OPM is a subset of PROV. There are some limitations on OPM: for example, OPM is not able to present concept of set and relationship between ***agent <– data*** and ***agent <– agent***, and also OPM is not able to describe the concept of plan (the set of steps and actions executed by an agent for certain targets).

There are some other related works such as label flow framework for annotating workflow provenance ([32]), semantics study for the Taverna 2 workflow model ([33]) and using Taverna as a tool for building and running workflows of services ([34]). There are providing more insights on provenance data in scientific workflow but is not structured to address the data model in business workflow.

Continuous evolvement occurs in research on data provenance, model and the applications, including works in [10] and [35]. In particular, Enabling Privacy in Provenance-Aware Workflow Systems Enabling Privacy in Provenance Aware Workflow Systems ([10]) studies how to use provenance data to ensure privacy in workflow. This helps to answer how data privacy can be ensured at workflow design time.

In anther major work from Davidson, A Propagation Model for Provenance Views of Public / Private Workflows ([35]) studies two different types of workflow: public and private and how they propagate in a certain workflow specification context. The visibility and reachability during the data propagation in both public and private workflows are investigated. More works (2016, 2017) in this area studies and contribute to the following: Composition and substitution [36], and Workflow with Heterogeneous event [37]. There are other similar works proposed partially different solutions. In [38], a regular path query algorithm is proposed for workflow provenance; in [39], a labelling method for run time recursive workflow execution is introduced; in [40], a method to estimate provenance for complex application is introduced; in [41], [42], and [29]), a series of evolution on Open Provenance Model is introduced; in [43], an association methodology is introduced to bridge workflow and data provenance, called 'Strong Links'. In [44], the general philosophy of workflow provenance research for business workflow is introduced, but not in details how to query workflow provenance with respect to future executions and past executions.

In next section, related work in workflow patterns and queries are be explored.

2.3 Workflow Patterns and Queries

From literature, query on workflow is investigated predominantly on provenance model.

general query and query language in provenance study In general, if a workflow is represented by a graph, the query is typically from a given set of nodes and edges to the dependent nodes and edges in the workflow. Here terms of nodes are typically representing data inputs or outputs and edges are representing operations or activities. The nodes could be initial nodes (with no incoming

edges), intermediate nodes (with incoming edges and outgoing edges) or final nodes (with no outgoing edges). In term of query language, typical SQL can be found in Zoom ([45]) which is based on Oracle DB. Another query language is user defined SPAROL which is introduced in View ([46]). As these are general queries, they require user to be familiar on the underlying data model, otherwise it is easy to make mistakes in the queries. There are some data provenance systems providing a set of API for user to use, such as Taverna which provides three layers of API: RDF API, core API and analytics API. Some other systems provide user defined query such as VisTrails ([11]) and Kepler ([31]). They are rich in capability, but limited to the the particular systems only. OPQL ([47]) is a query language for OPM model, which provides 6 basic graph models for user to efficiently query recursively. OPQL also provides four operations on sub-graph, including abstract, add, intersect, and minus.

Workflow Reachability and Labelling Solution In major research work, workflow can be described in a precise and structured grammar and language so that closer look and analytics on the data (data provenance) becomes achievable. The provenance concept evolves from classic definitions on workflow using Petri net, BPNM, etc. ([3]). This thesis takes reference on the related work on the data and trace log in scientific workflow and business workflow, in order to resolve data inconsistency issue.

Later theories on how data should be annotated and transferred are set up by researchers such as Tova Milo ([9]), Davidson Susan ([10]), and Freire Juliana ([11]). Data provenance models and the polynomial operators (semi-rings) are introduced in [12]. These are dated from year 1998 to year 2011. Also some challenges in provenance study are summarised in [48] and [49].

One of the related work on solving data dependencies between workflow modules is reachability. Reachability is basically to study the dependencies between input or output of a module and input or output of another module. With the foundation in workflow and data provenance theory, study on workflow model and solution for particular questions such as data reachability is examined in the work from Bao [13] and others. These are dated from year 2012 to year 2014.

The author in [13] uses a fine-grained workflow dependencies (refer to Figure 2.7) which provides visibility on dependencies between outputs and the exact depending inputs. This is an advance from previous assumption on coarse-grained dependencies (refer to Figure 2.7) or called black box.

A common algorithm is provided in Bao's work to calculate reachability between inputs and outputs of workflow, including any internal workflow modules, as shown in Figure 2.6. Two notes are to

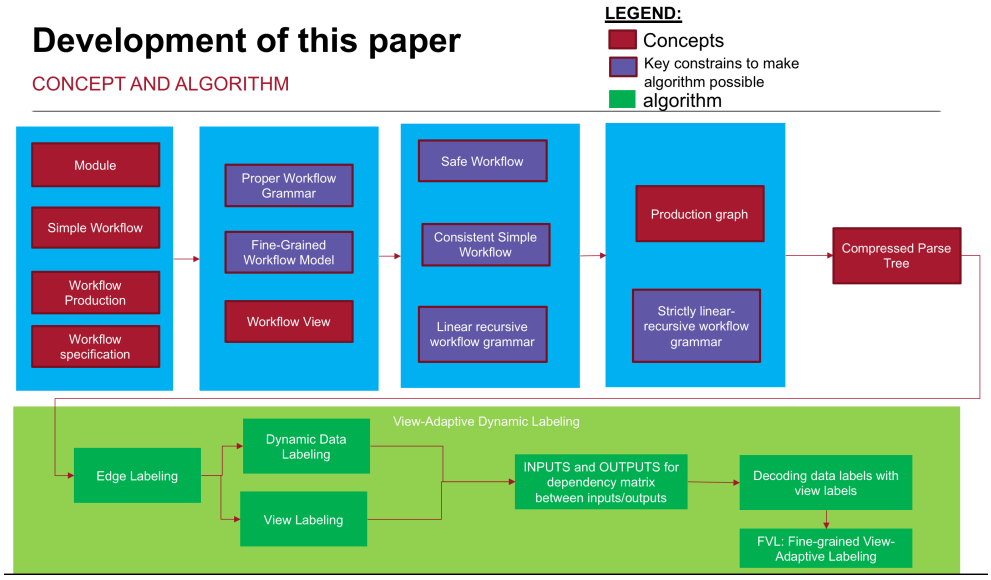


FIGURE 2.5: summary of definitions, constraints and algorithms development in Bao's paper [13]

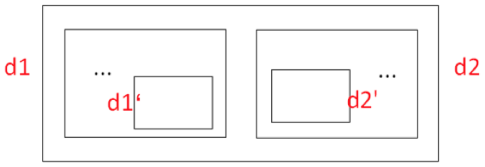


FIGURE 2.6: illustration of inputs and outputs in workflows (reference from [13])

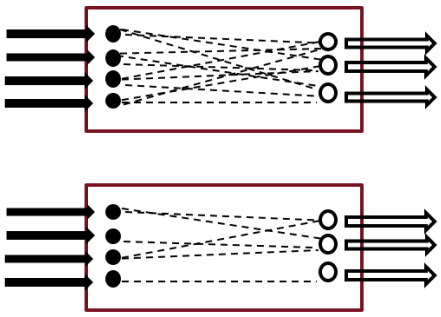


FIGURE 2.7: a sample coarse-grained dependency and fine-grained dependency (from [13])

be considered: 1) the input could be initial input (d_1) or intermediate input (d_1´); the output could be final output (d_2) or intermediate output (d_2´); 2) initial input and final output concept to any workflows (such as d_1 and d_2).

As shown in the tree diagram (Figure 2.8 on 21), if there are more and more recursive executions in the workflow, the tree depth will become bigger and bigger and go out of control eventually. This will cause the calculation time unpredictable. The strictly linear recursive workflow grammar restricts that there is no joint in any recursive workflow, refer to Figure 2.9. Based on all of the above definitions and restrictions, the author proves that the compressed parse tree (shown in Figure 2.8) is logarithmic in length and the calculation on reachability can be achieved in constant

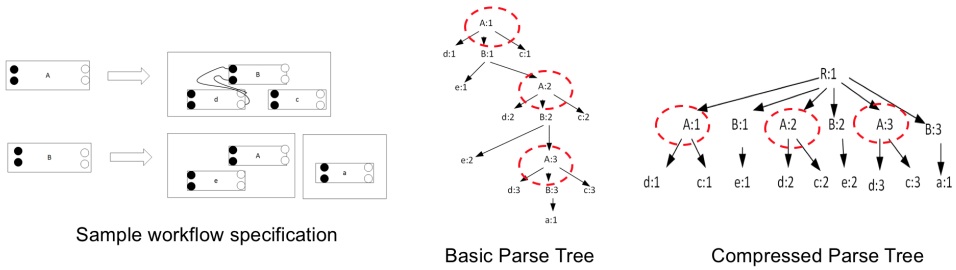


FIGURE 2.8: Basic Parse Tree and Compressed Tree

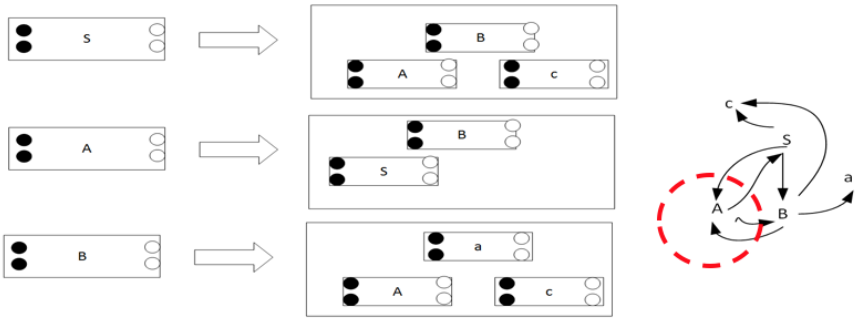


FIGURE 2.9: a sample workflow which is not strictly linear recursive

time.

2.4 Discussion

After analysing the related works described in previous chapters, there is still a gap on how to model data and workflow provenance in business workflow and how to examine reachability and traceability in complex data modelling. The key limitation is that the reachability solution and other similar work does not deal with specific characteristics of business workflow, including multiple starting point, dynamic execution and dependencies on temporal behaviours and data synchronisation between workflow modules.

The reachability query in previous section can not resolve the problem described in section 2.1. There are the following key limitations identified in the reachability approach in workflow data provenance:

1. data type is simple value (single value as shown in Figure 2.6);
2. the workflow model does not consider multiple initial states;
3. there's no consideration on dependencies in data mode;
4. there's no consideration on temporal dependencies.

In summary, the related work has two major gaps: 1) not able to express data inconsistency situation with comprehensive semantics; 2) lack of query algorithm to find out data inconsistency. These limitations impose negative impacts on the capabilities in applying data provenance to business workflow data inconsistency problem. In this work, advantages of both BPMN and data provenance are utilised in building the description model and solution algorithm. The following aspects are also taken into consideration:

complex data model In business workflow, data is complex through inheritance and aggregation. Some data model shares common set of attributes from other data model through inheritance. Some data model is fulfilled only when all the sub data model aggregated.

multiple initial workflow modules In business workflow, there are many sub systems and roles which could initiate new processes. Some business workflow processes can also be initiated by events.

design time check It will be beneficial to business if potential issue in data consistency can be identified at design time of workflow specification. This requires simulation on the workflow run with different conditions and identification on the potential data inconsistency issue.

task execution pattern When data consistency occurs at certain point of time in workflow execution, it will be beneficial to business if there is a way to trace backwards and find exactly which dependencies were missed out, and what tasks can be executed as compensation.

consideration on temporal dependencies As emphasised in the example, one major factor of the data inconsistency problem is that a particular task may depend on data to be generated by another task in future. Although there is temporal description in related work, it is still lacking of actionable analysis.

This thesis is not intending to address all the limitations. The BPMN model will be enhanced to include additional information on data. A data inconsistency query algorithm is developed during the thesis project, with enhanced definition of workflow module, by introducing data hierarchy, data states, roles and time. This forms a foundation for future work. In addition, category of data inconsistency is discussed in section 3.3.

3

Methodology

The data inconsistency solution in this thesis consists of the following three components:

1. Time and Data based BPMN (TD-BPMN), an enhanced BPMN modelling on business processes with details of activity, data, states, state transitions and temporal properties
2. category of data consistency identified in business processes
3. algorithm to query data inconsistency from a give business process defined in TD-BPMN

In this chapter, preliminaries are first introduced which will be referred in the model and algorithm.

3.1 Preliminaries

Before elaborating the modelling and query algorithm, the semantics definitions are first specified for workflow, workflow module, workflow specification, FSM, and graph grammar. These are the foundations in research and solution investigation.

Definition 3.1.1. *(**Role**) A role (R) is a human being or system who is participating in a business workflow.*

Definition 3.1.2. *(**Activity**) An activity (A) represents the fact that a role takes operations on data objects. The operations include read, write, create and delete.*

Definition 3.1.3. *(**Workflow Module**) A workflow module is one node in a business workflow where an activity is taking place. Sometimes activity is referred by a workflow module for simplicity.*

Example 1. *In Figure 3.2 on page 28, $a1$ is a workflow module.*

Definition 3.1.4. *(**Edge**) Edge is the directional linkage between two workflow modules. Edge represents the execution of the business workflow from A end to B end.*

Example 2. *In Figure 3.2 on page 28, the arrow between workflow module $a1$ and $a2$ are Edges.*

Definition 3.1.5. *(**Workflow**) Workflow is a set of {A,E,D,S,R,T }, where*
A is a set of Activities.
E is a set of Edges.
D is a set of data model.
S is a set of states.
R is a set of Roles.
T is a set of temporal properties. Refer to Definition 3.1.10 for all types of temporal properties.

Definition 3.1.6. *(**BPMN graph**) BPMN stands for Business Process Modelling Notation. It is a graphical representation of workflow. The categories and elements can be found in Figure 2.1 on page 13, cited from [3].*

Example 3. *One sample BPMN graph is shown in Figure 3.2 on Page 28.*

Definition 3.1.7. *(**Workflow Run**) Workflow run, is a list of ordered activities as result of execution according to workflow specification.*

Workflow run can also be represented by graph. A sample graph is shown in right side in Figure 3.1 with respect to the workflow specification on the left.

Definition 3.1.8. *(**Workflow Run Grammar**) Workflow Run Grammar is a set of {a}, where*
$\exists$ Workflow $W = \{A,E,D,S,R\}$
$\forall$ ab, $a,b \in A$, and $X(a,b)=1$, X is a temporal operator and refer to Definition 3.1.10 for all temporal operators.

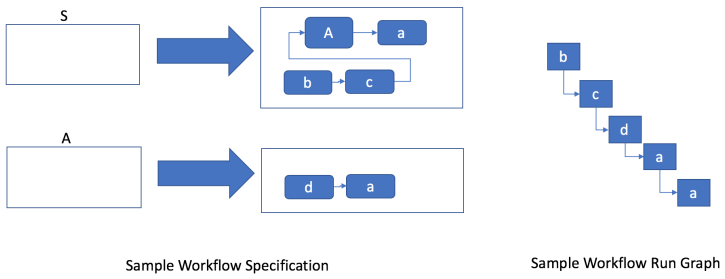


FIGURE 3.1: Sample Workflow Specification and Workflow Run graph

Example 4. *The workflow execution shown in Figure 3.1 can be represented by a languages of bcdaa.*

Definition 3.1.9. *(**Finite State Machine**) Finite State Machine (FSM) is a set of {A,E,D,S,R}, where S is set of states,*
$\forall\ s,\ s \in S$
E is set of edges between states,
$\forall\ e,\ e \in E$ *and e:* $s1 -> s2$
A is set of activities associated with edge,
$\forall\ a,\ a \in A$

Example 5. *A sample FSM is shown in Figure 3.3. In the sample FSM, symbols such as an are referring to activities. For instances,* $a1$ *and* $a2$ *represents the associated activities in the transition from initial device state to 'Created & Provisioned' state.*

Definition 3.1.10. *(**Workflow Temporal Properties and Notation**) Workflow Temporal Properties enhances Workflow definition with Temporal information. And Workflow Temporal Notation is specific notations on top of BPMN diagram representing the Properties.* $WTN=\{A,Min,Max,X\}$*, where:*
Min *is the mapping of A with minimum duration:*
$Min:\ a \rightarrow n,\ \forall\ a \in A$
Max *is the mapping of A with maximum duration:*
$Max:\ a \rightarrow n,\ \forall\ a \in A$
X *is the function to determine whether one activity must happen after another activity:*
$X:\ a,b \rightarrow bb,\ bb \in \{0,1\}$ *Note: There time is defined in natural number. As per Theorem 3.1.1, using natural number is sufficient to define workflow order and temporal sequence.*

Example 6. *Here is a example. of a workflow with 3 activities.* $W=\{a1,a2,a3\}$.
$Min(a1)=1$
$Max(a1)=3$

$Min(a2)$=1
$Max(a2)$=3
$Min(a3)$=1
$Max(a3)$=3
X: $a1$,$a2 \rightarrow 1$
X: $a3$,$a2 \rightarrow 1$

Theorem 3.1.1 (**Activity Duration**). *Workflow activity duration can be represented using natural number in design so that duration of any two activities can be compared.*

Proof. In a given workflow specification, number of activities are finite and a set can be defined as $\{a1,...,aN\}$ to represent all activities, and a set $\{d1,...,dN\}$ to represent durations of all activities. $\exists$ L so that 1×10^{L}>N and $1 \times 10^{L-1}$<=N, $\forall\ i$, $di \in \{d1,...,dN\}$, rewrite the duration as:

$$\lfloor di \rfloor \times 10^{L} + i$$

The converted duration is unique for all activities and is natural number. □

Definition 3.1.11. *(**Commutative Condition**) Two activities are commutative if they are reading from different attributes of same object or from different objects, or if they are updating to different attributes of same object or from different objects.*

Theorem 3.1.2 (**Commutative Enforcement**). *Two activities are commutative if and only if they are reading from different attributes of same object or from different objects, or updating to different attributes of same object or from different objects.*

Proof. The **if** part is true as per definition. Contradiction is used to prove the **only if** part. If two activities are reading from same attributes, the value could be different from different order and it may impacting the object states and final workflow execution, thus the two activities are not always commutative. If two activities are updating to same attributes, the updated value may be different in different order and it may impact the object states and the final workflow execution path, thus the two activities are not commutative. With the contradictory analysis, the only if stays. □

Lemma 1. *If two activities a and b in a workflow specification are commutative, the state after ab and ba are same.*

Definition 3.1.12. *(**Commutative Order**) For any list of activities, any order by only changing the execution sequence of activities which are commutative is called commutative order.*

Theorem 3.1.3 (**State Transition Condition**). *An object transfers from state Sa to State Sb if and only if the state transition activities are executed in commutative order.*

Remark 1. *The Theorem 3.1.3 is an extension from Theorem 3.1.2 and Lemma 1 and is the foundation of the algorithms introduced in next section.*

3.2 TD-BPMN Model

In order to reveal the data operated in workflow activities, FSM and UML class diagram is added as enrichment to BPMN diagram. The full set of descriptive diagrams is called Time and Data enriched BPMN (TD-BPMN).

Definition 3.2.1. ***(TD-BPMN)*** *TD-BPMN stands for Time and Data enriched BPMN. TD-BPMN describes a set of business processes with three diagrams, including BPMN, FSM and UML Class Diagram. Where:*

- *BPMN keeps original Business Process Model and Notation (described in chapter 2.1) with add-on of Temporal Properties Notation (in definition 3.1.10). Expected states are annotated to related activities, events or gateways when applicable to indicate that the activity, event or gateway can only proceed with correct outcome when the states are met as entry condition. Minimum and maximum durations are annotated to each activity, event and gateway.*
- *UML Class Diagram describes data object model involved in BPMN activities. The involvement includes CRUD (Create, Read, Update and Delete) operations on whole data object and attributes of data object.*
- *FSM describes states of data object and the state transition edges. For each state transitions, a list of activities are listed on the edge. The state transition needs to follow State Transition Condition rule in Theorem 3.1.3.*
- *at run time, tokens (represented by circle graphically) are added in the enriched BPMN to indicate workflow execution step.*

Example 7. *A simplified example of TD-BPMN is shown in Figure 3.2, 3.3 and 3.4. This serves as a practical modelling on the research problem and focuses on the major data inconsistency issue.*

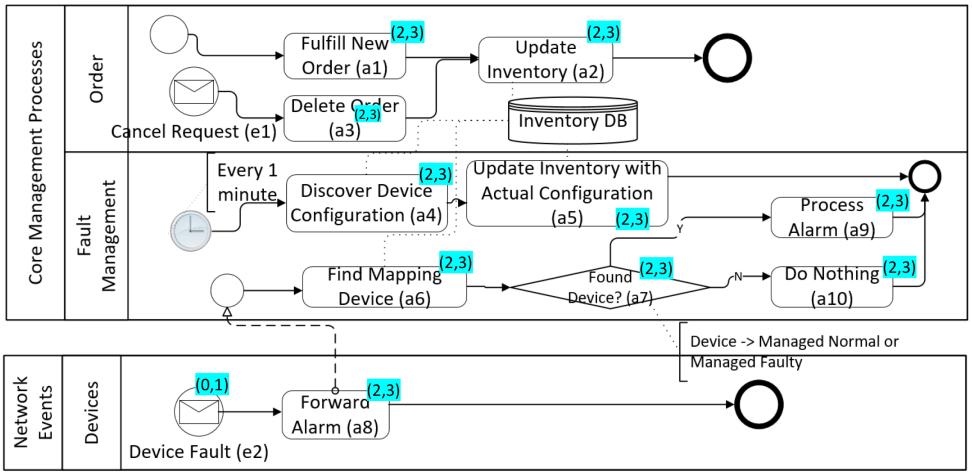


FIGURE 3.2: TD-BPMN: BPMN definition of order, inventory, fault management and device networks

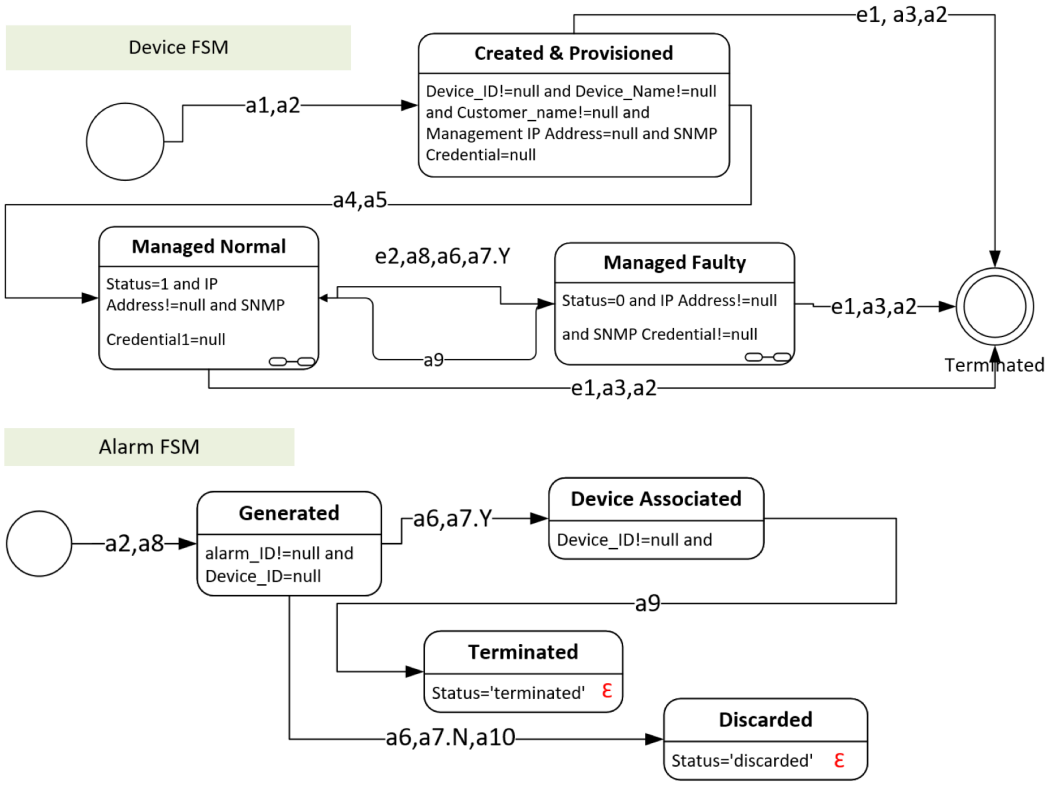


FIGURE 3.3: TD-BPMN: Finite State Machine of Device and Alarm

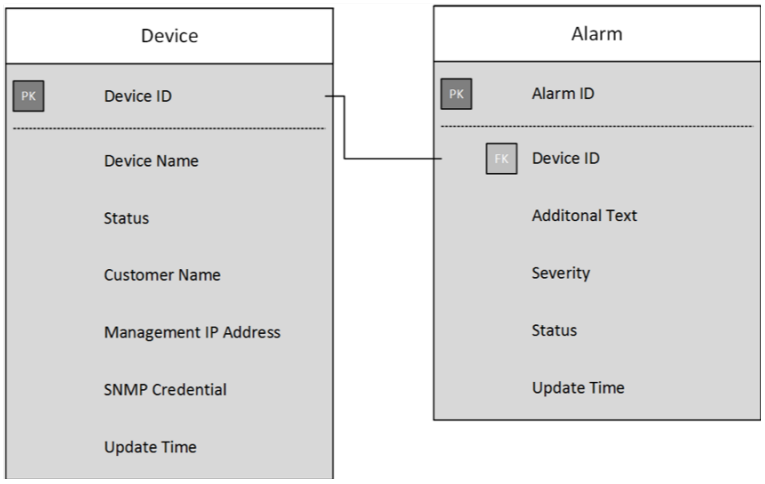


FIGURE 3.4: TD-BPMN: UML Class Diagram of Device and Alarm

The sample BPMN diagram (Figure 3.2) contains two pools and 3 lanes. One pool is for $Core\ Management\ Process$ which includes $Fault\ Management$ and $Order$ lanes. Another pool is $Network\ Events$ and includes Devices lane only. Temporal properties are added in to BPMN diagram. Minimum and maximum durations on each activity and event is indicated.

In first lane, $Order$, there are two starting activities, $i1$ and $e1$. $i1$ is a general initial activity as a starting point of a normal new order process. $e1$ is triggered from a *Cancel Request* from customer. As the request can come in at any time, it is modelled as an event rather than initial activity. *Update Inventory* activity will either create the new entry of service and devices or delete the existing service and devices in inventory database. In second lane, *Fault Management*, there are two starting activities, one scheduled event $e3$ and one *Alarm Event* $e4$. The $e3$ is reoccurring at a predefined interval of 1 minute which triggers *Discover Device Configuration* ($a4$) and *Update Inventory with Actual Configuration* ($a5$). $a4$ checks whether there are any addition or deletion in the inventory database, and $a5$ updates the changes in inventory system to *Fault Management* lane local repository for fast processing purpose. $e4$ is an event triggered by one *Forward Alarm* activity in *Network Events* pool. $e4$ leads to an alarm handling procedure which tries to resolve the fault indicated by alarm through *Process Alarm* if the alarm is associated with a device, or *Do Nothing* ($a10$) if the alarm can not be associated with any device. Expected State annotation is added on *Found Device?* which stands for that the gateway is expecting the device to be in either *Managed Normal* or *Managed Faulty* state before returning correct result. The last lane, *Devices*, starts with a *Device Fault* ($e2$) event and performs *Forward Alarm* ($a8$) activity which forwards alarm to *Fault Management* lane.

The UML Data Model diagrams in Figure 3.4 contains two data object model. *Device* contains one

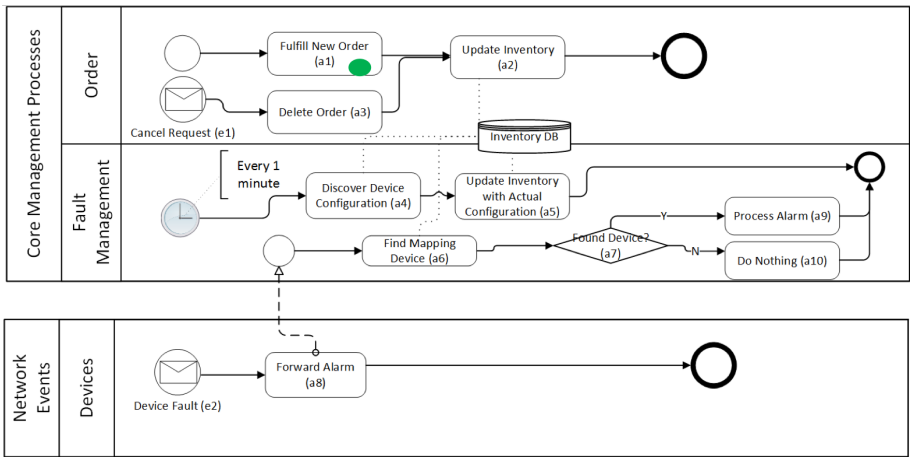


FIGURE 3.5: BPMN token in Execution Run

Device ID as primary key, and other attributes such as *Device Name* and *Status*. *Alarm* contains *Alarm ID* as primary key and *Device ID* as foreign key for association with device, as well as other attributes such as *Additional Text* and *Status*.

The FSM diagram in Figure 3.3 contains Finite State Machine diagrams for $Device$ and $Alarm$. There are 3 states for Devices apart from initial and terminated states, *Created & Provisioned*, *Managed Normal* and *Managed Faulty*. The activities marked on each state transition edges are the state transition triggering condition. For instance, the device can only enter normal operational state (*Managed Normal*) after *Discover Device Configuration* ($a4$) and *Update Inventory with Actual Configuration* ($a5$) activities. In each state, there is also a marked on attributes condition. For instance, only when *State=1* and *Management IP Address* and *SNMP Credential* are both **null**, the device state is *Managed Normal*. *Alarm* object is defined in similar way. The ϵ indicates a terminating state as long as an *Alarm* instance enters *Terminated* or *Discarded* state.

By tracing the workflow execution with combination of BPMN and FSM, the full picture is unleashed. For instance, Figure 3.5 and 3.6, a snapshot of workflow execution and states of key objects are illustrated. The snapshot is at 10:05 AM, the business workflow is executed up to node $a1$, *Fulfil New Order*. The device is at *Created and Provisioned* state. There are two device instances created. The first device has most of attributes assigned, but the second device has not been configured thus the *Management IP address* and *SNMP credential* are left empty. From the FSM, it is clearly demonstrated that the two fields will be filled when the device enters *Managed Normal* or *Managed Faulty* states.

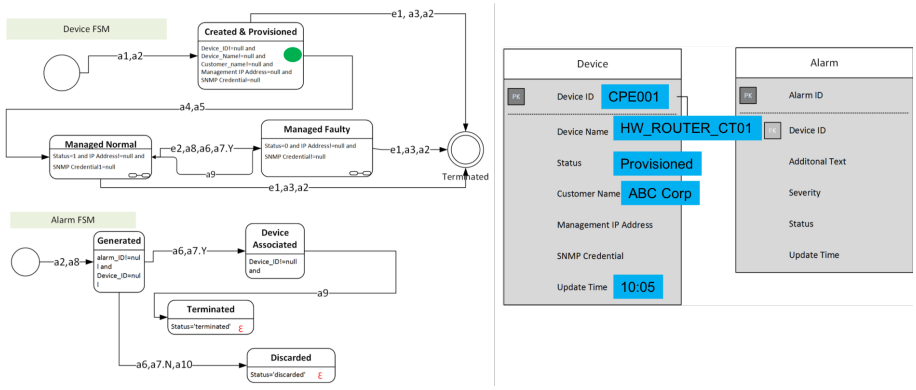


FIGURE 3.6: Instance View in Execution Run

The workflow execution can be described step by step. Due to the limitation of thesis length, the execution steps are not listed one by one. The execution snapshot is highlighted at the point when data inconsistency occurs. As shown in Figure 3.7, there are two concurrent threads running in Order process group (called pool), one is deleting an existing order at $a3$, the other is creating a new order at $a1$. At the same time, there an device fault alarm coming to the system, and the handling process is up to $a7$. The FSM and UML status of the workflow is shown in Figure 3.8, where there are two device instances. CPE001 is at Deleted status, and CPE002 is at Normal status. There is also an alarm instance, however, as it is just created, there is only information generated from the device, including IP address in the additional text. The CPE002 is assigned same IP address as the IP address was released from CPE001 after deletion. The alarm instance is generated from CPE002 and containing the IP address as 10.10.23.3. The data inconsistency issue occurs at this point of time. In *Found Device?* activity, the system trying to find associated device with the IP Address in Alarm instance. As the delete device CPE001 is still providing complete information in local inventory, and the newly created device CPE002 has not yet been assigned an IP address and been synchronised into *Fault Management* local inventory, the alarm matching activity associated the alarm with CPE001 rather than CPE002. The *Process Alarm* activity will be triggered as next step as the system is trying to resolve the fault status on the device which includes dispatching a truck roll and human resource (technician) to fix the CPE001. The technician will in the end find out that the alarm is linked with a non-existing device as it was already deleted. The monetary cost on the truck roll and technician round trip is a complete loss to the telecommunication company. Imagining that the system is a running system and there could be considerable amount of new devices created and old device being deleted, flooding with alarms, the loss could be accumulating quickly.

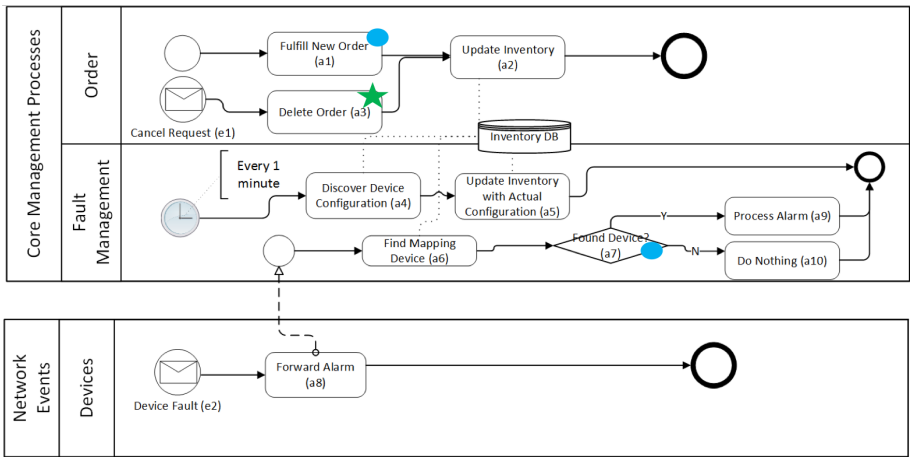


FIGURE 3.7: BPMN view when data inconsistency occurs

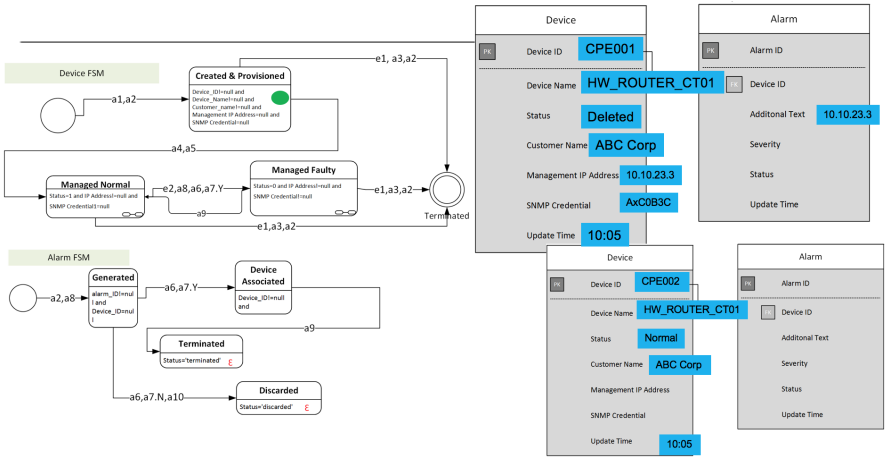


FIGURE 3.8: FSM and Model instance view when data inconsistency occurs

3.3 Classification of Data Inconsistency

With the definition of TD-BPMN in 3.2, the key elements are activity, data and time. The different combinations of the elements may result in potential data inconsistency problems. One typical situation is an activity depends on data to be generated by another activity in advance. As shown in Figure 3.9, there are two sample activities for a mobile network operation. Activity *capacityPlan* is to plan whether there is a need to expand network equipment and activity *recordUsage* generates network consumption data. *capacityPlan* depends *consumption* data. If *capacityPlan* starts before *recordUsage* even finishes, the *capacityPlan* activity will face data inconsistency issue and provides a misleading capacity plan.

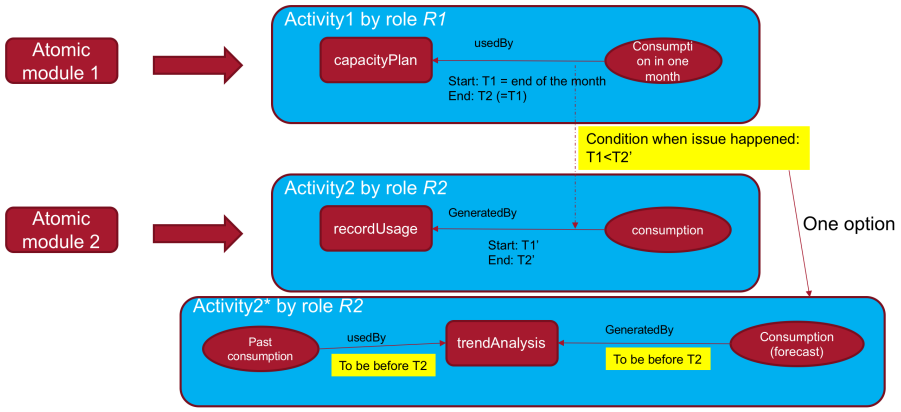


FIGURE 3.9: One Data Inconsistency Category

The classification of the data inconsistency categories identified in the research are listed as follows:

1. Depends on a data which will be generated in future
2. Depends on a data which will be updated in future
3. Depends on a data which will be deleted in future
4. Depends on a data which was happened by few runs in the past
5. Depends on an activity which will happen in future
6. Depends on a role activated (with any activity) in future
7. Depends on a set of data in both past and future
8. Depends on a state transfer in data, start in the past and end in future
9. Depends on a state transfer in data, start and end in past
10. Depends on a state transfer in data, start and end in future
11. Depends on a set of data which should be retrieved at same time

The classification is based on analysis of real life business processes (telecommunication management reference in [50] and network configuration reference in [51]). Examples for each pattern are listed below. Telecommunication examples are used but the patterns can be applied to other business domains.

P1: depends on a data which will be generated in future

The sample explained above when capacity planning action depends on network consumption being generated. If capacity planning occurs before all consumption data is generated, it is a future dependence violation. Figure 3.9 illustrates the pattern.

P2: depends on a data which will be updated in future

As per the research problem in section 1.2, it depends on device IP address being correctly updated to process alarm in right way.

P3: depends on a data which will be deleted in future

In a network service provision activity, engineer needs to assign network interface to a service. If a designed interface is not freed up from anther service, the engineer needs to wait until the interface is released.

P4: depends on a data which was happened by few runs in the past

Engineer will issue a device investigation only when the CPU utilisation of the devices went up 80% 4 times in the past.

P5: depends on an activity which will happen in future

Similar as P1, *capacityPlan* activity depends on *recordUsage* to be completed.

P6: Depends on a role activated (with any activity) in future

As shown in Figure 1.2 in section 1.2, there are different roles in the telecommunication management systems. In particular, *Customer Care* depends on *Service Management* to dispatch correct trouble tickets to fix any network problems in customer's network. Before *Customer Care* functions properly, *Service Management* role needs to be in place.

P7: Depends on a set of data in both past and future

For a particular network equipment, the equipment faulty is defined by four types of alarms. At a particular point of time, if three types of alarms have already been generated, the faulty identification activity will depends on the fourth which could occur in future.

P8: Depends on a state transfer in data, start in the past and end in future

A device problem is specified as CPU Utilisation goes beyond 80% 4 times in 10 minutes. If a problem check activity starts when there were 3 time occurrences in the past 5 minutes, the activity

depends on 4th occurrence.

P9: Depends on a state transfer in data, start and end in past

For the CPU utilisation scenario described in P7, operator will only mark the device as normal when CPU utilisation goes beyond 80%. The start and end of 80% status needs to be completed before the clearance of the device issue.

P10: Depends on a state transfer in data, start and end in future

To predict there is a need to introduce new network equipment device, a prediction needs to be done to examine whether a CPU utilisation 80% fault status will be entered in future.

P11: Depends on a set of data which should be retrieved at same time

Network devices are controlled by multiple roles, to ensure accuracy of network configuration report for a sub network, all attributes of the devices need to be retrieved at same time in an atomic task.

This thesis does not provide resolutions on all categories due to research time constraint. A query algorithm is developed to identify data inconsistency issue for a given workflow specification described in TD-BPMN.

3.4 Query Algorithm and Implementation

3.4.1 First Attempt

Query algorithm is developed and tested for identifying data inconsistency. Limitations are found in the first attempt algorithm and an improved version is developed to overcome the limitations. The first attempt algorithm is introduced in this subsection as a reference point.

The overall algorithm of first attempt is shown in Figure 3.10 which checks data consistency for a given activity with respect to a pre-designed workflow specification. Two threads on the algorithms are running in parallel and then merge for comparison. The thread on the left is to find out activity list leading to the expected state of the activity; the thread on the right is to find out all possible workflow run leading to the given activity, which also takes into consideration on the temporal constrains on the activities as per workflow specification and expands the leading activity list with

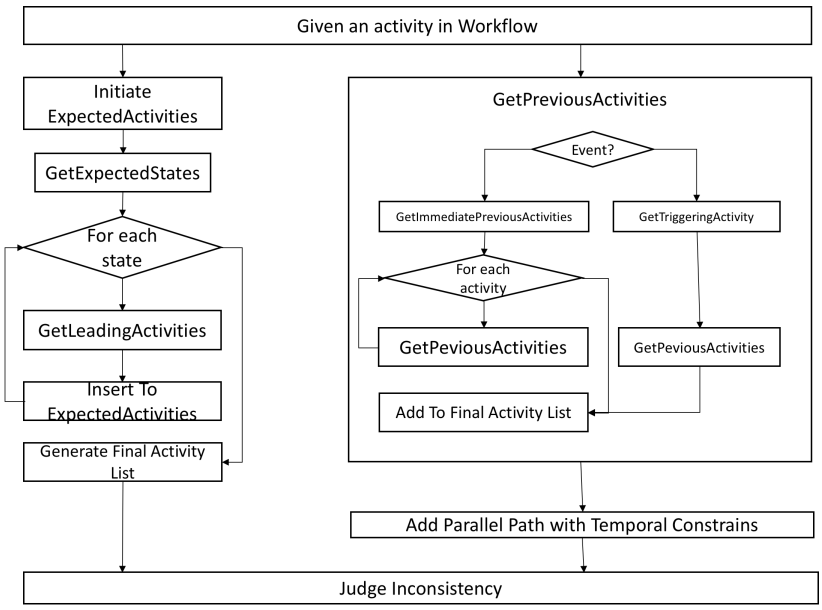


FIGURE 3.10: Overall Data Inconsistency Check Algorithm (first attempt)

all possible parallel activities. The algorithm is also described in algorithm 1 on Page 37.

The procedure for parallel path under temporal constrains is described in Figure 3.11. The constrains of non-scheduled events, scheduled events and duration of activities are taken into consideration in generating the possible leading activity list. The final judgment procedure ensures that any expected state in the given activities is derived by at least one possible workflow run path.

The following limitations are identified on the first attempt algorithm: 1), initial action could be called multiple times with multiple instances manipulated. for simplicity, the algorithm only considers up to two immediate occurrences on the initial actions; 2), in this thesis, commutative features are only considered between up to two activities. An improved algorithm is further investigated and implemented, which considers multiple runs and activities are traversed with all possible commutative relationship explored.

3.4.2 Improved Algorithm

From the algorithm described in previous subsection, the calculation of possible designed activity is too large, and they are not really required to make justification for data consistency. In real scenario, workflow run starting with a pure event is unpredictable as the event can happen anytime. If any

Algorithm 1 Inconsistency Check

```
1: procedure INCONSISTENCYCHECK(()a)▷ check whether there is data inconsistency possibility
   for the given activity a
2:     ExpectedActivity ← ∅
3:     states ← GetExpectedStates(a)
4:     while for each state ∈ states do
5:         activities ← GetLeadingActivities(state)
6:         ExpectedActivity ← activities
7:     matchingExpectedActivity ← GetPreviousActivites(a)
8:     ExpectedActivity ← filterTemporal(ExpectedActivity)
9:     matchingExpectedActivity ← filterTemporal(matchingExpectedActivity)
10:    inconsistency ←
11:    JudgeInconsistency(ExpectedActivity,matchingExpectedActivity)
12: procedure GETEXPECTEDSTATE(a)                    ▷ Get expected state of an activity a
13:    read activity design meta data Objects
14:    while  for each object ∈ Objects do           ▷ all objects operated by the activity
15:        push object.state
16:    return b                                      ▷ expected state list
17: procedure GETIMMEDIATEPREVIOUSACTIVITIES(()a)
18:    activities ← ∅
19:    while for edges ∈ E do
20:        if edge.right == a
21:        push edge.left to activities
22:    return activities                             ▷ expected state list
23: procedure GETPREVIOUSACTIVITIES(()a)              ▷ recursively get previous activities
24:    activities ← GetImmediatePreviousActivites(a)
25:    while for each activity ∈ activities do
26:        push GetPreviousActivites(activity) to activities
27:    return activities
```

state transition depends on a pure event generation, the execution sequence is not guaranteed thus there is risk in data inconsistency. As shown in Figure 3.12, the execution time of $e1$ can be either before $e3$, between $e3$ and $a4$, or after $a4$, there are no consistency guarantee from the workflow pool starting with $e1$.

With this consideration, an improved version is developed which removed the unnecessary consideration on event execution. The improved algorithm is shown in Figure 3.13.

The key improvements from first attempt algorithm (in Figure 3.10) are: 1) *Add Parallel Path with Temporal Constrains* is removed from the first attempt algorithm, this reduces significantly the amount of redundant calculation; 2) temporal logic check is added to the final inconsistency

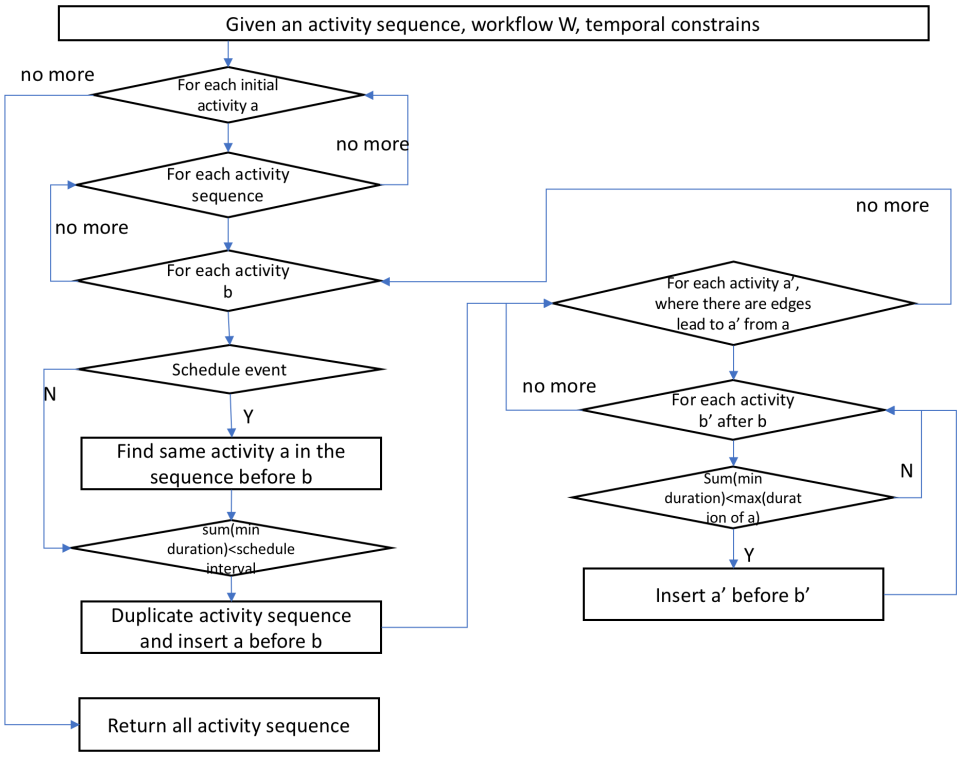


FIGURE 3.11: Procedure for parallel path with temporal constrains

judgment step in the improved algorithm. Instead of exploring all possible workflow run, temporal properties of events and activities are examined and consistency status in one state can be identified directly.

3.5 Summary

In this chapter, definitions are first introduced to specify workflow, activity, temporal specifications on activity. Three Theorems are set up as the foundation for inconsistency query algorithm.

TD-BPMN model is introduced to provide in-depth descriptive language on business processes so that data inconsistency problem can be explored semantically. A classification on data inconsistency is then analysed. An algorithm is developed to solve the data inconsistency query problem.

In next chapter, case study result is analysed with the complexity of workflow run and efficiency in identifying data inconsistency by the query algorithm.

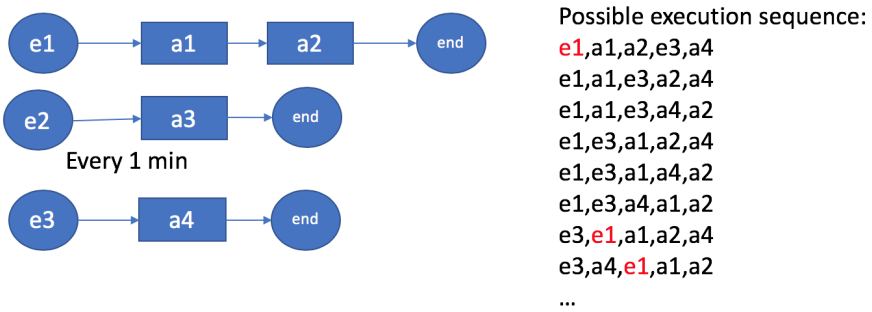


FIGURE 3.12: Sample execution sequence with pure event

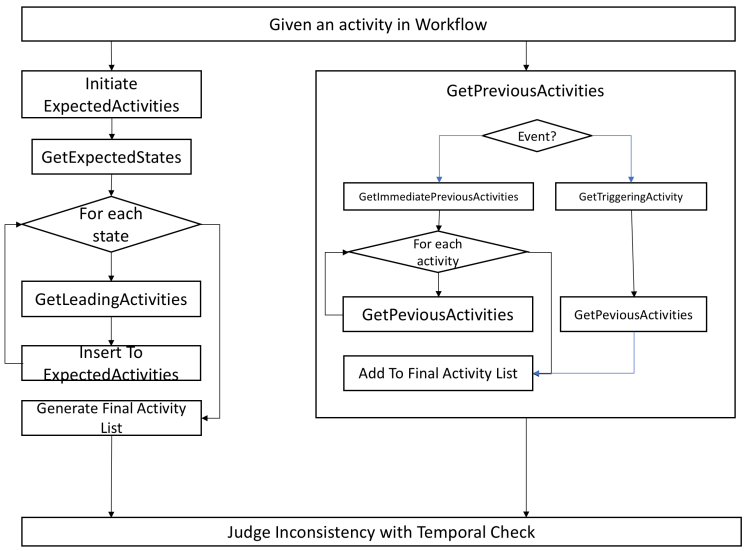


FIGURE 3.13: Improved High Level Algorithm

4

Case Study and Results

In this chapter, query algorithm will be examined with simulated business process definitions. The execution durations of the algorithm are measured agains number of business process pools and activities.

4.1 Case Study Overview

To verify the efficiency and accuracy of the algorithm, two case studies are conducted.

First case study is to simulate possible workflow execution path with a given workflow specification. The workflow specification is given in comma-separated values definition. Two types of definitions are given, including BPMN and FSM. In BPMN, activities and edges between activities are given. For each activity, the expected states are also given as well as the expected minimum and maximum duration of the activity. In FSM, objects states and the transition activities are given. Sample definitions are shown below.

BPMN comma-separated values file:

$activity, i0, , 2, 3$
$activity, a1, , 2, 3$
$activity, a2, device.s1, 4,$
$edge, i0, a1,$
$edge, a1, a2,$
$edge, a2, a3,$
FSM comma-separated values file:
$device, start, s1, i0$
$device, s1, s2, a1 :: a2$
$device, s2, s3, a3 :: a4$

In BPMN input definition, rows starting with $activity$ are for all activity involved in the business workflow, and in the example, $i0$ is an initial activity and $a1$ is one activity; rows starting with $edge$ are for edges, for example, there is one edge from $i0$ to $a1$. In FSM input definition, the format is *object class*,*state origin*,*state destination*,*action list*. For action list, the state is only allowed to be transited if and only if the actions are executed in order, for example, to transit device form state $s1$ to state $s2$, activities $a1$ and $a2$ must be completed in order. The BPMN input definition provides minimum and maximum duration with last two columns on the rows starting with activity, and empty stands for undetermined. In the above example, $i0$ has a minimum duration of 2 and maximum duration of 3; $a2$ has a minimum duration of 4 and undetermined maximum duration. As per Theorem 3.1.1, natural number is used to specify the duration, and the mapping with actual real life duration is to be done separately.

Number of activity nodes and number of workflow pools are the two aspects used in the experiment. As per literature review, workflow run depth is related with number of activities. And to examine the effect of temporal logic impact between workflows, the number of pools is another important factor. The input workflow specifications is examined with 10, 20, 300 activity nodes, in 2,3,4,5 and 10 pools. The generation of workflow specification samples took references of some real workflow in the telecommunication industry. For experiment purpose, a workflow specification simulation is used to generate the different combination. The simulation takes number of activity nodes and number of pools as input, and assigns activity nodes evenly to pools, and assigns half of the pool starting with initial activity and another half starting with event.

Second case study is to verify the data inconsistency query algorithm. The expected object state are randomly assigned to activity for simulation purpose. The inconsistency calculation time taken is recorded for different number of activity noes, under different number of pools. For temporal

properties, it is added as constraints to the business workflow, including 1) an activity can not last longer than the duration which avoids an activity running too long in parallel with other activities; 2) for scheduled events, the occurrence is at predefined interval. The algorithm is examined for number of activity nodes up to 1,000, as per some sample real case business workflow, 1,000 is a reasonable sufficient setup to implement most of typical business processes.

The advantage of the experiment data generation method is:

1. able to represent full information of business workflow and finite state machine without loosing factors
2. flexible in examination on different possibilities in workflow design
3. flexible in integration with other main stream business workflow research and industry systems
4. efficient in eliminating human intervention on workflow elements selection
5. random in different workflow execution path patterns

There are other approaches in getting the experiment data, including 1) using real workflow as inputs; 2) design specific workflow for the experiment purpose; 3) replicate a small workflow by extending activity, edges, and states with same ratio. The reasons why the workflow simulation is adopted are as follows: For real workflow, the workflow may limit to particular application, and as the selection is a human decision process, it may loose some generality. Regarding option 2 specific design, this is again involving too much human intervention, and also time consuming and error proof when looking after huge amount of data. For the option 3 on workflow replication and expansion, as the ratio needs to kept same and it follows similar workflow execution pattern, the results may not be valuable for large amount of activity nodes and edges.

4.2 Workflow Execution Path Simulation

The possible of workflow run paths are first examined. Some assumptions are made in order to concentrate on the data inconsistency problem. Although in real world events can happen multiple times, to concentrate only on the data inconsistency, only up to 2 event occurrences are considered in the experiments. Another fact is that event or activity duration could vary, to concentrate only

on the data inconsistency, it is considered to be same in duration. Thirdly, gateway activity is assigned randomly in the simulation.

Figure 4.2 provides a high level view on how many possible workflow run on number of activity nodes and number of pools. Form the experiment, when number of activity nodes is greater and equal to 15 and number of pools is greater and equal to 3, the number of possible workflow runs grows exponentially, 1×10^4 is used an indication on this trend rather than actual value.

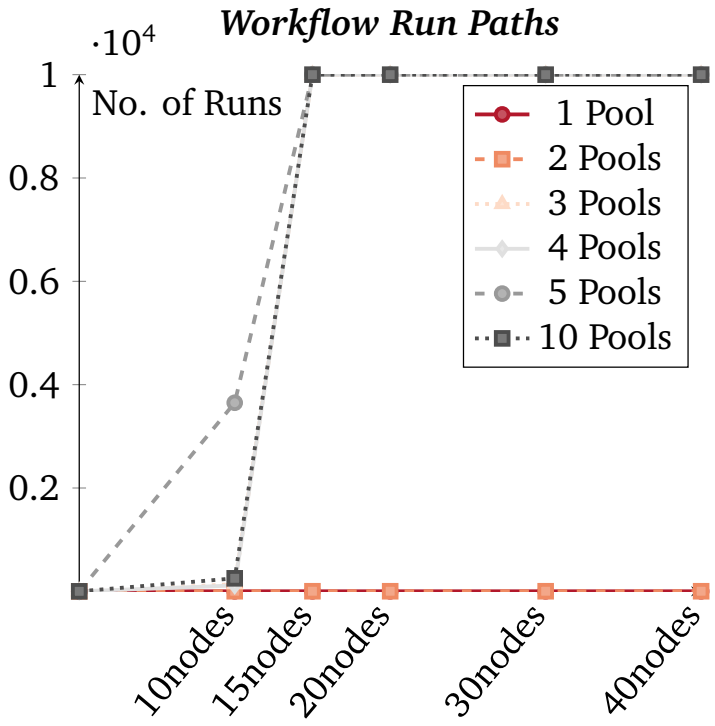


FIGURE 4.1: Possible workflow run paths for number of nodes in different pools

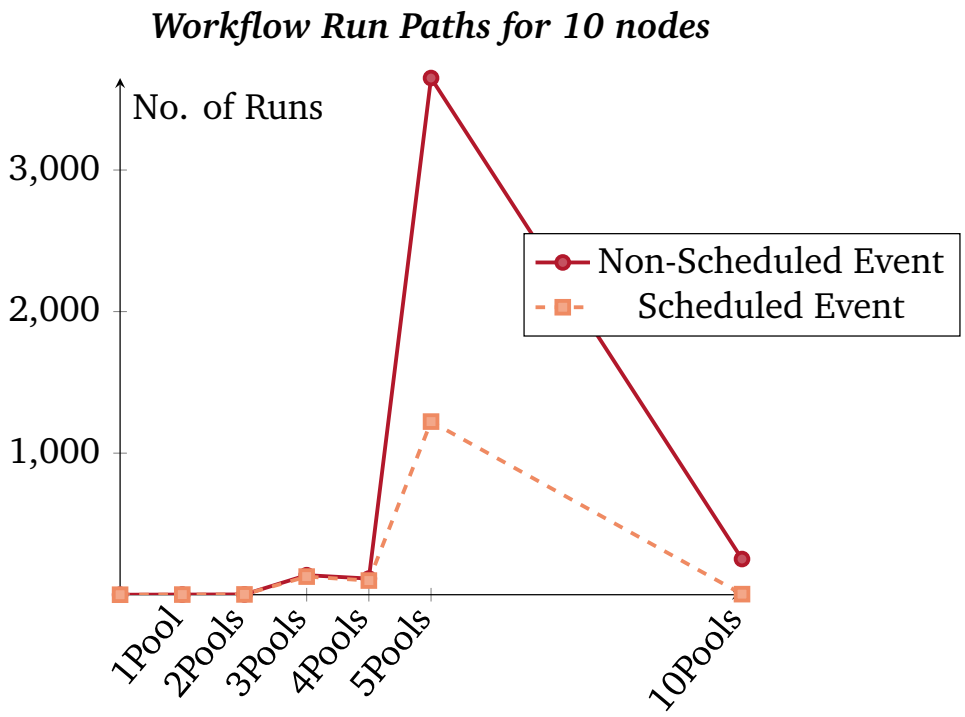


FIGURE 4.2: Possible workflow run paths with the difference on non scheduled and scheduled events

As next step, the workflow run paths for 10 nodes are examined under different number of pools and two category of events (non-scheduled event and scheduled event). As shown in Figure 4.2, although there are not much difference when the number of pools is less and equal to 4, overall number of possible workflow run paths is significantly low when workflow specification is based on scheduled event comparing with non-scheduled event. Considering the real life temporal nature of business processes, in the implementation of the data inconsistency query algorithm, process events are considered as scheduled events only.

4.3 Algorithm Results

The improved algorithm (shown in Figure 3.13) is implemented in Java, in a Mac Pro with following specifications: CPU, Intel(R) Core(TM) i7-4770HQ CPU @ 2.20GHz; Memory, 16 GB.

The algorithm takes workflow specification with temporal properties and FSM as inputs and provides the following results:

1. data inconsistency result for each and every activity node

2. the gap activity sequence to make the data back to consistency
3. totally how many activity nodes are in consistency and how many in inconsistency

The algorithm is executed for number of nodes up to 1,000 under number of pools of 1, 2, 3, 4, 5 and 10. The time taken in the algorithm execution is shown in Figure 4.3 and Figure 4.4 on Page 45 and 46. X-axis is the number of nodes, Y-axis is algorithm execution time in milliseconds.

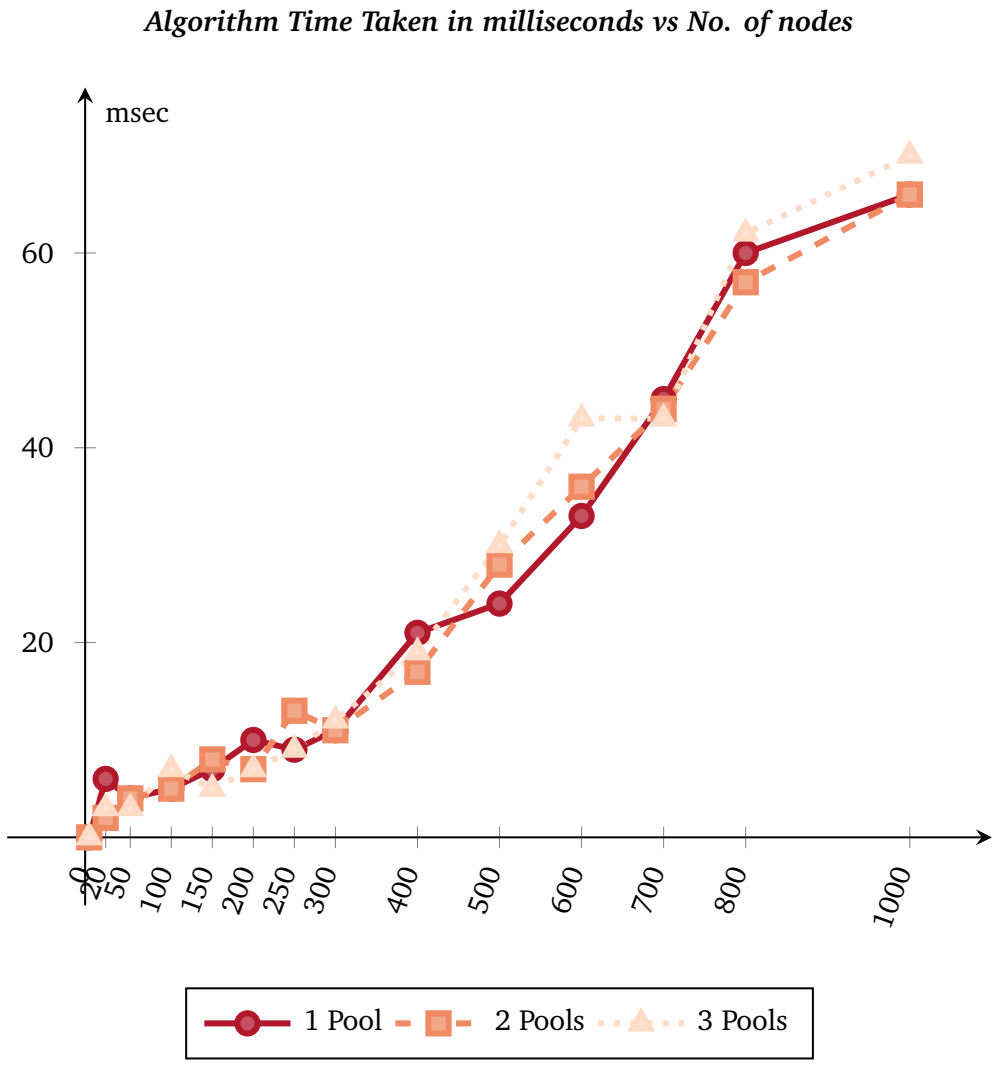


FIGURE 4.3: Inconsistency Check Algorithm Performance (1-3 Pools)

The reason why up to 1,000 nodes are examined is explained in previous chapter. The results are further discussed in the next chapter.

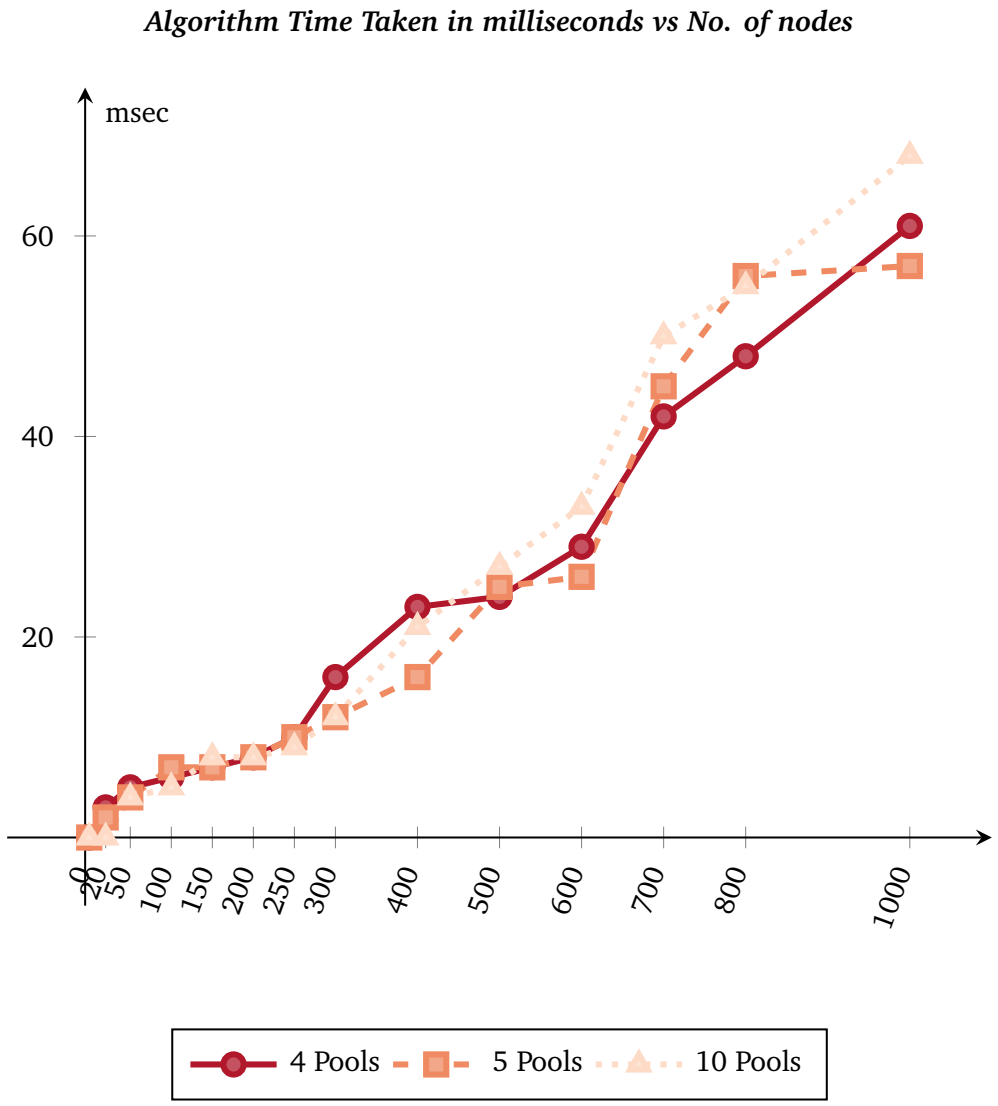


FIGURE 4.4: Inconsistency Check Algorithm Performance (4,5,10 Pools)

4.4 Discussion

Case study experiments results will be analysed and discussed in this chapter.

Findings

As shown in Figure 4.2 on Page 43, when there are only one or two pools, the number of possible runs are relatively minimized. The reason is that the activities can only happen on sequence in one pool case and in alternative order when there are two pools. However, when there are more than 2 pools, the number of possible workflow runs grow exponentially when there are more than 15 nodes. The reason is that there are multiple entry into the workflow, and the activity may float

between the pools, which make the combination grow exponentially.

In Figure 4.2, number of possible runs is substantially smaller with scheduled events than non-scheduled events. for instance, in 5 pools situation, non-schedule events situation leads to more than 3000 possible execution paths, but scheduled events situation only leads to 1000 plus execution paths. Scheduled events are happening in predefined interval, with workflow execution occurring as per the specification following initial activities, scheduled events happen in accordance with fixed number of main flow execution activities. From the result, the algorithm is concluding results in a consistent time.

The conversion from TD-BPMN design to inputs data to the algorithm is straightforward. The mapping form workflow module to activity is one to one, same for workflow execution path to edge, data state into FSM state. According to the result, the execution time taken in the algorithm is not exponential, although the possible number of workflow execution paths is growing exponentially per Figure 4.2.

Case Study Limitations

Due to the time constrain in this work, the experiment has two major limitations listed below: 1) the experiment assumes same and fixed duration of all activities; 2) the experiment adopts a fix ratio between initial activities and events.

The hypothesis is that the results will be relatively similar as the experiment result presented in this thesis. From duration point of view, it may impact activity execution sequence. However, in most situation, if two activities execution order can revere, it implies that they do not depend on each other. In most situation, the dependency between two activities is on data. Independent activities mean the read and write on different object or different attributes. As per Theorem 3.1.2, the two activities are commutative thus it does not change the inconsistency check result. From workflow design and the activity event ratio point of view, the algorithm is examining execution path following workflow specification. Regardless of the number of initial activities and event, workflow execution will follow same rule and the algorithm needs to go through workflow activities and it is a linear relationship.

Formal proof and analysis will be conducted in future work.

5

Conclusion and Future Work

In this chapter, the research work is concluded and ideas on future work are explored.

5.1 Research Summary

The research targets to find a solution for data inconsistency issue in business processes.

By reviewing related work, two major research areas are identified: BPMN and Data Provenance. As data manipulation and investigation is the core in workflow data provenance approach, similar resolution on data reachability is explored. The approach resolves partially the data inconsistency problem, as explained in chapter 3.

As per workflow execution simulation results shown in section 4.2, the number of possible workflows execution paths surges when there are more than 15 activities and more than 3 pools. This makes it difficult for workflow designer to identify potential data inconsistency issue.

This thesis provides a data inconsistency solution with the following three contributions:

Time and Data enhanced BPMN (TD-BPMN)

TD-BPMN describes a set of business processes using three inter-related diagrams, including BPMN, UML Class Diagram and Finite State Machine. The main objective of introducing TD-BPMN is to enrich BPMN diagram with insights on data model view and how data states are transited over time. As discussed in section 1.1, although BPMN diagram provides details on sequence and message flows between activities in business processes flow, it does not provide details on how data manipulated between activities. The UML Class Diagram defines all data object classes involved in business process including class attributes. Finite State Machine (FSM) diagram provides relationships between BPMN and UML Class Diagram. FSM contains two major aspects. One aspect is the state of each and every data object. In each state, there are value conditions specified for data object attributes. Anther aspect is the activities notations for each state transit, which stands for the triggering condition of the state transit is the occurrence of the series of activities in oder; second notification is to define minimum and maximum duration as temporal properties on each activity.

In addition, there is two more notations added to BPMN, for each activity in BPMN, if the activity relies on certain object to be in specific state, the state will be marked as a expected state notation to the activity; and also minimum and maximum duration of the activity is another type of annotation. The TD-BPMN provides comprehensive description of business processes, including 1) how many activities in the business workflow, how they are organised and transferred; 2) for each activity, what are the entry condition; 3) how many data objects manipulated in the business workflow, how they are transited between states and what are the triggering condition of the data object state transition. This addresses the first problem in the problem statement in section 1.2.3.

Classification of data inconsistency categories

As per the TD-BPMN model, activity, data and time are major properties of a business process. The also forms different categories of data inconsistency. For instance, one activity may depend on a data object which should be destructed in future time.; if the data object is not deleted at a specific time in future, the current activity is not providing correct outcome and data inconsistency problem occurs. Details of the data inconsistency categories are provided in section 3.3. This addresses the second problem in the problem statement in section 1.2.3.

Business Process Query identifying data inconsistency

The business process query takes input of a business processes described in TD-BPMN, and identifies the activities with potential data inconsistency possibility. The query first examines the expected states of each activity and the sequence of activities leading to the states. Secondly, the query

checks in the possible workflow runs, whether there are workflow run-time execution sequence aligning with the activity sequence identified in first step. The algorithm filters workflow run paths with temporal properties to reduce the number of workflow run instances and ensure timely completion. Performance of the query algorithm is studied and indication of a $O(n)$ complexity is discussed. This addresses the last problem in the problem statement in section 1.2.3.

5.2 Future Work

Data consistency problem in business workflow is a significant issue from both technical and financial point of view. Based on the inconsistency query algorithm, the research work on the data inconsistency will be continued.

Future work includes 1) proving on the non-exponential nature of the algorithm and further optimisation on the algorithm to cater for even large scale workflow design; 2) enrich further temporal properties associated with major elements in BPMN; 3) formal modelling on categories of data inconsistency identified in this thesis; 4) study on the comprehensive solution for dealing with data inconsistency problem. In the complete solution, after the data inconsistency query, workflow execution pattern will be examined for each activity with data inconsistency potential, and a workflow execution path will be extended to ensure data consistency by ensuring the expected data model states for data consistency.

Bibliography

[1] E. Newcomer and G. Lomow. *Understanding SOA with Web Service* (Pearson Education, 2005).

[2] W. M. P. van der Aalst and K. M. van Hee. *Workflow Management: Models, Methods, and Systems* (MIT Press, 2002).

[3] M. Weske. *Business Process Management: Concepts, Languages, Architectures* (Springer, ISBN 978-3-540-73521-2 Springer Berlin Heidelberg New York, 1998).

[4] C. A. Petri. *Communication with Automata*. Applied Data Research **15**(8), 357 (1966).

[5] W. M. P. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. J. M. M. Weijters. *Workflow mining: A survey of issues and approaches*. Data Knowl. Eng. **47**(2), 237 (2003). URL `https://doi.org/10.1016/S0169-023X(03)00066-1`.

[6] M. Fowler and K. Scott. *UML Distilled Second Edition A Brief Guide to the Standard Object Modeling Language* (Wesley Publisher Addison, 1999).

[7] H. Thomas and E. James. *The New Industrial Engineering: Information Technology And Business Process Redesign*. Sloan Management Review, Summer **31**(4), 11 (1990).

[8] J. Su. *Dynamic constraints and object migration*. Theor. Comput. Sci. **184**(1-2), 195 (1997). URL `https://doi.org/10.1016/S0304-3975(96)00142-9`.

[9] Y. Amsterdamer, S. B. Davidson, D. Deutch, T. Milo, J. Stoyanovich, and V. Tannen. *Putting lipstick on pig: Enabling database-style workflow provenance*. CoRR **abs/1201.0231** (2012). `1201.0231`, URL `http://arxiv.org/abs/1201.0231`.

[10] S. B. Davidson, S. Khanna, V. Tannen, S. Roy, Y. Chen, T. Milo, and J. Stoyanovich. *Enabling privacy in provenance-aware workflow systems*. In *CIDR 2011, Fifth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 9-12, 2011, Online Proceedings*, pp. 215–218 (www.cidrdb.org, 2011). URL `http://cidrdb.org/cidr2011/Papers/CIDR11_Paper30.pdf`.

[11] C. E. Scheidegger, H. T. Vo, D. Koop, J. Freire, and C. T. Silva. *Querying and re-using workflows with vstrails*. In J. T. Wang, ed., *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pp. 1251–1254 (ACM, 2008). URL `https://doi.org/10.1145/1376616.1376747`.

[12] T. J. Green, G. Karvounarakis, and V. Tannen. *Provenance semirings*. Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems - PODS '07 (June), 31 (2007). URL http://portal.acm.org/citation.cfm?doid=1265530.1265535.

[13] Z. Bao, S. B. Davidson, and T. Milo. *Labeling workflow views with fine-grained dependencies*. PVLDB **5**(11), 1208 (2012). URL http://vldb.org/pvldb/vol5/p1208_zhuoweibao_vldb2012.pdf.

[14] E. Bazhenova, F. Zerbato, and M. Weske. *Data-centric extraction of DMN decision models from BPMN process models*. In E. Teniente and M. Weidlich, eds., *Business Process Management Workshops - BPM 2017 International Workshops, Barcelona, Spain, September 10-11, 2017, Revised Papers*, vol. 308 of *Lecture Notes in Business Information Processing*, pp. 542–555 (Springer, 2017). URL https://doi.org/10.1007/978-3-319-74030-0_43.

[15] H. Arthur, V. D. Aalst, and W. M. Yawl. *YAWL : Yet Another Workflow Language (Revised version)*. Information Systems **30**, 245 (2005).

[16] A. Yousfi, M. Hewelt, C. Bauer, and M. Weske. *Towards uBPMN-Based Patterns for Modeling Ubiquitous Business Processes*. IEEE Transactions on Industrial Informatics **1**(December), 99 (2017).

[17] E. Bazhenova and M. Weske. *Optimal acquisition of input data for decision taking in business processes*. Proceedings of the Symposium on Applied Computing - SAC '17 pp. 703–710 (2017). URL http://dl.acm.org/citation.cfm?doid=3019612.3019615.

[18] S. Mandal, M. Weidlich, and M. Weske. *Events in business process implementation: Early subscription and event buffering*. Lecture Notes in Business Information Processing **297**, 141 (2017).

[19] S. Mandal, M. Hewelt, and M. Weske. *A framework for integrating real-world events and business processes in an IoT environment*. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **10573 LNCS**(October), 194 (2017).

[20] T. Baier, C. Di Ciccio, J. Mendling, and M. Weske. *Matching events and activities by integrating behavioral aspects and label analysis*. Software and Systems Modeling **17**(2), 573 (2018).

[21] V. D. Aalst. *Business process management as the " Killer App " for Petri nets*. Software & Systems Modeling (2015). URL http://dx.doi.org/10.1007/s10270-014-0424-2.

[22] I. V. Stetsenko. *Parallel Algorithm for Petri Object Simulation*. Cybernetics and Systems Analysis **53**, 605 (2017). URL https://doi.org/10.1007/s10559-017-9963-1.

[23] M. L. Rosa, W. M. P. van der Aalst, M. Dumas, and F. Milani. *Business process variability modeling: A survey*. ACM Comput. Surv. **50**(1), 2:1 (2017). URL https://doi.org/10.1145/3041957.

[24] G. Eugene, P. Zhao, L. Di, A. Chen, M. Deng, and Y. Bai. *Computers & Geosciences BPELPower - A BPEL execution engine for geospatial web services*. Computers and Geosciences **47**, 87 (2012). URL http://dx.doi.org/10.1016/j.cageo.2011.11.029.

[25] R. K. L. Ko. *A Computer Scientist's Introductory Guide to Business Process Management (BPM)*. Crossroads XRDS **15**(4), 11 (2009). URL http://delivery.acm.org/10.1145/1560000/1558901/p11-ko.pdf?ip=147.86.207.243{&}id=1558901{&}acc=OPEN{&}key=FC66C24E42F07228.89BA465B3F8E8007.4D4702B0C3E38B35.6D218144511F3437{&}CFID=448174072{&}CFTOKEN=14616714{&}{_}{_}acm{_}{_}=1398768135{_}525e27b6e3f50dd837c6d4c3c3b866d6.

[26] E. Deelman, G. Singh, M.-h. Su, J. Blythe, Y. Gil, and C. Kesselman. *Pegasus : A framework for mapping complex scientific workflows onto distributed systems* **13**, 219 (2005).

[27] K. Wolstencroft, R. Haines, D. Fellows, A. Williams, D. Withers, S. Owen, S. Soiland-reyes, I. Dunlop, A. Nenadic, P. Fisher, J. Bhagat, K. Belhajjame, F. Bacall, A. Hardisty, A. Nieva, D. Hidalga, M. P. B. Vargas, S. Sufi, and C. Goble. *The Taverna workflow suite : designing and executing workflows of Web Services on the desktop , web or in the cloud* **41**(June), 557 (2018).

[28] H. Park, R. Ikeda, and J. Widom. *RAMP: A system for capturing and tracing provenance in mapreduce workflows*. PVLDB **4**(12), 1351 (2011). URL http://www.vldb.org/pvldb/vol4/p1351-park.pdf.

[29] L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. T. Groth, N. Kwasnikowska, S. Miles, P. Missier, J. Myers, B. Plale, Y. Simmhan, E. G. Stephan, and J. V. den Bussche. *The open provenance model core specification (v1.1)*. Future Generation Comp. Syst. **27**(6), 743 (2011). URL https://doi.org/10.1016/j.future.2010.07.005.

[30] L. Carata, S. Akoush, N. Balakrishnan, T. Bytheway, R. Sohan, M. Seltzer, and A. Hopper. *A primer on provenance*. Commun. ACM **57**(5), 52 (2014). URL https://doi.org/10.1145/2596628.

[31] I. Altintas, O. Barney, and E. Jaeger-Frank. *Provenance collection support in the kepler scientific workflow system* **4145**, 118 (2006). URL https://doi.org/10.1007/11890850_14.

[32] P. Alper, K. Belhajjame, V. Curcin, and C. A. Goble. Label*flow framework for annotating workflow provenance*. Informatics **5**(1), 11 (2018). URL https://doi.org/10.3390/informatics5010011.

[33] J. Sroka, J. Hidders, P. Missier, and C. Goble. *A formal semantics for the Taverna 2 workflow model*. Journal of Computer and System Sciences **76**(6), 490 (2010). URL http://dx.doi.org/10.1016/j.jcss.2009.11.009.

[34] D. Hull, K. Wolstencroft, R. Stevens, C. A. Goble, M. R. Pocock, P. Li, and T. Oinn. *Taverna: a tool for building and running workflows of services*. Nucleic Acids Research **34**(Web-Server-Issue), 729 (2006). URL https://doi.org/10.1093/nar/gkl320.

[35] S. B. Davidson, T. Milo, and S. Roy. *A propagation model for provenance views of public/private workflows*. ICDT pp. 165–176 (2013). URL https://doi.org/10.1145/2448496.2448517.

[36] P. Buneman, A. Gascón, and D. Murray-Rust. *Composition and substitution in provenance and workflows*. TaPP (2016). URL https://www.usenix.org/conference/tapp16/workshop-program/presentation/buneman.

[37] Y. Gao, S. Song, X. Zhu, J. Wang, X. Lian, and L. Zou. *Matching heterogeneous event data*. IEEE Trans. Knowl. Data Eng. **30**(11), 2157 (2018). URL https://doi.org/10.1109/TKDE.2018.2815695.

[38] X. Huang, Z. Bao, S. B. Davidson, T. Milo, and X. Yuan. *Answering regular path queries on workflow provenance*. In J. Gehrke, W. Lehner, K. Shim, S. K. Cha, and G. M. Lohman, eds., *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*, pp. 375–386 (IEEE Computer Society, 2015). URL https://doi.org/10.1109/ICDE.2015.7113299.

[39] Z. Bao, S. B. Davidson, and T. Milo. *Labeling recursive workflow executions on-the-fly*. In T. K. Sellis, R. J. Miller, A. Kementsietsidis, and Y. Velegrakis, eds., *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June 12-16, 2011*, pp. 493–504 (ACM, 2011). URL https://doi.org/10.1145/1989323.1989375.

[40] E. Ainy, S. B. Davidson, D. Deutch, and T. Milo. *Approximated provenance for complex applications*. In A. Chapman, B. Ludäscher, and A. Schreiber, eds., *6th Workshop on the Theory and Practice of Provenance, TaPP'14, Cologne, Germany, June 12-13, 2014* (USENIX Association, 2014). URL https://www.usenix.org/conference/tapp2014/agenda/presentation/ainy.

[41] L. Moreau, J. Freire, J. Futrelle, R. E. Mcgrath, J. Myers, and P. Paulson. *The Open Provenance Model (v1.00)*. Technical report, University of Southampton (2007). URL http://eprints.ecs.soton.ac.uk/14979.

[42] L. Moreau, J. Freire, J. Futrelle, R. E. McGrath, J. Myers, and P. R. Paulson. *The open provenance model: An overview*. In J. Freire, D. Koop, and L. Moreau, eds., *Provenance and Annotation of Data and Processes, Second International Provenance and Annotation Workshop, IPAW 2008, Salt Lake City, UT, USA, June 17-18, 2008. Revised Selected Papers*, vol. 5272 of *Lecture Notes in Computer Science*, pp. 323–326 (Springer, 2008). URL https://doi.org/10.1007/978-3-540-89965-5_31.

[43] D. Koop, E. Santos, B. Bauer, M. Troyer, J. Freire, and C. T. Silva. *Bridging workflow and data provenance using strong links*. In M. Gertz and B. Ludäscher, eds., *Scientific and Statistical Database Management, 22nd International Conference, SSDBM 2010, Heidelberg, Germany, June 30 - July 2, 2010. Proceedings*, vol. 6187 of *Lecture Notes in Computer Science*, pp. 397–415 (Springer, 2010). URL https://doi.org/10.1007/978-3-642-13818-8_28.

[44] D. Deutch and T. Milo. *A Quest for Beauty and Wealth (or , Business Processes for Database Researchers)*. PODS '11 Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems pp. 1–12 (2011).

[45] O. Biton, S. Cohen-boulakia, and S. B. Davidson. *Zoom * UserViews : Querying Relevant Provenance in Workflow Systems [Demonstration Proposal]* pp. 1366–1369 (2007).

[46] C. Lin, S. Lu, Z. Lai, A. Chebotko, X. Fei, J. Hua, and F. Fotouhi. *Service-oriented architecture for VIEW: A visual scientific workflow management system*. In *2008 IEEE International Conference on Services Computing (SCC 2008), 8-11 July 2008, Honolulu, Hawaii, USA*, pp. 335–342 (IEEE Computer Society, 2008). URL https://doi.org/10.1109/SCC.2008.118.

[47] C. Lim, S. Lu, A. Chebotko, and F. Fotouhi. *OPQL: A first opm-level query language for scientific workflow provenance*. In H. Jacobsen, Y. Wang, and P. Hung, eds., *IEEE International Conference on Services Computing, SCC 2011, Washington, DC, USA, 4-9 July, 2011*, pp. 136–143 (IEEE Computer Society, 2011). URL https://doi.org/10.1109/SCC.2011.60.

[48] L. Moreau, B. Ludäscher, I. Altintas, R. S. Barga, S. Bowers, S. P. Callahan, G. C. Jr., B. Clifford, S. Cohen, S. C. Boulakia, S. B. Davidson, E. Deelman, L. A. Digiampietri, I. T. Foster, J. Freire, J. Frew, J. Futrelle, T. Gibson, Y. Gil, C. A. Goble, J. Golbeck, P. T. Groth, D. A. Holland, S. Jiang, J. Kim, D. Koop, A. Krenek, T. M. McPhillips, G. Mehta, S. Miles, D. Metzger, S. Munroe, J. Myers, B. Plale, N. Podhorszki, V. Ratnakar, E. Santos, C. E. Scheidegger, K. Schuchardt, M. I. Seltzer, Y. L. Simmhan, C. T. Silva, P. Slaughter, E. G. Stephan, R. Stevens, D. Turi, H. T. Vo, M. Wilde, J. Zhao, and Y. Zhao. *Special issue: The first provenance challenge*. Concurrency and Computation: Practice and Experience **20**(5), 409 (2008). URL https://doi.org/10.1002/cpe.1233.

[49] S. B. Davidson and J. Freire. *Provenance and scientific workflows: challenges and opportunities*. In J. T. Wang, ed., *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pp. 1345–1350 (ACM, 2008). URL https://doi.org/10.1145/1376616.1376772.

[50] M. B. Kelly. *The telemanagement forum's enhanced telecom operations map (etom)*. J. Network Syst. Manage. **11**(1), 109 (2003). URL https://doi.org/10.1023/A:1022449209526.

[51] Y. Wang, E. Keller, B. Biskeborn, J. E. van der Merwe, and J. Rexford. *Virtual routers on the move: live router migration as a network-management primitive*. In V. Bahl, D. Wetherall, S. Savage, and I. Stoica, eds., *Proceedings of the ACM SIGCOMM 2008 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Seattle, WA, USA, August 17-22, 2008*, pp. 231–242 (ACM, 2008). URL https://doi.org/10.1145/1402958.1402985.