WORKFLOW BEHAVIOR AUDITING FOR MISSION CENTRIC COLLABORATION

A Dissertation

by

JOHN MATTHEW PECARINA

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Jyh-Charn Liu
Committee Members,	Joobin Choobineh
	Guofei Gu
	Frank Shipman
Head of Department,	Nancy Amato

December 2013

Major Subject: Computer Science

ABSTRACT

Successful *mission-centric* collaboration depends on situational awareness in an increasingly complex mission environment. To support timely and reliable high level mission decisions, auditing tools need real-time data for effective assessment and optimization of mission behaviors. In the context of a battle rhythm, mission health can be measured from workflow generated activities. Though battle rhythm collaboration is dynamic and global, a potential enabling technology for workflow behavior auditing exists in process mining.

However, process mining is not adequate to provide mission situational awareness in the battle rhythm environment since event logs may contain dynamic mission states, noise and timestamp inaccuracy. Therefore, we address a few key near-term issues. In sequences of activities parsed from network traffic streams, we identify mission state changes in the workflow shift detection algorithm. In segments of unstructured event logs that contain both noise and relevant workflow data, we extract and rank workflow instances for the process analyst. When confronted with timestamp inaccuracy in event logs from semi automated, distributed workflows, we develop the flower chain network and discovery algorithm to improve behavioral conformance. For long term adoption of process mining in mission centric collaboration, we develop and demonstrate an experimental framework for logging uncertainty testing. We show that it is highly feasible to employ process mining techniques in environments with dynamic mission states and logging uncertainty.

Future workflow behavior auditing technology will benefit from continued algorithmic development, new data sources and system prototypes to propel next generation mission situational awareness, giving commanders new tools to assess and optimize workflows, computer systems and missions in the battle space environment.

DEDICATION

I dedicate this dissertation foremost to God, who I prayed to daily to supply my needs and help me graduate. It is not his will that we walk without victory in our lives, but that victory is at his choosing. In 2001, he granted me everlasting Victory in his son Jesus Christ, who covers my sin and makes me whole. This grace he gave to complete this dissertation pales in comparison to the grace he gave to rescue my soul from death, yet it is a testament to his unbounded love that I am graced all the more in this day of completion.

He also graced me with family. Marcela is my soul mate and nothing could be done in my life if she was not the woman she is, supporting me and challenging me to be better. She remains faithful and true to me as a picture of Christ. From our union, we received four blessings whose names are a testimony of the spiritual journey of my doctoral study and my life: Samuel, Victoria, Evangeline and Nathanael. Samuel means that God hears, Victoria reminds me that God gives victory, Evangeline reminds me that we must tell the world and Nathanael, who was born the month of my preliminary exam, carries a reminder in his name that God provides. While at Texas A&M, I appreciated my time I could spend with my parents and parent in-laws; they were very supportive and encouraging. To my wife, children, parents, and parents in law, I dedicate the fruits of this labor.

ACKNOWLEDGEMENTS

I thank my friends, Josh and Amy White, who allowed me to live in their home for the final stretch of dissertation writing. I thank them for their good example of discipline and faith. I give many thanks to Mike George and Tim Nix, with whom I struggled alongside. Tim was an easy friend to talk to and exchange ideas with and set the bar high for me. Mike gave me so much support that I can never repay, even if he let me.

I acknowledge my research partners in this endeavor. The United States Air Force, through the Air Force Institute of Technology, sponsored my research work. I also thank my committee. Dr. Mac Lively sadly passed away shortly after my prelim, but in his final professional act he pushed me forward. I am grateful for Dr. Shipman who stepped up in his place. Dr. Gu was my initial contact and a reason for coming to Texas A&M and he gave me great advice in the prelim that I am grateful for. Last, but not least, Dr. Liu, who has become more and more a friend since I met him, has extended my ability for professorship immeasurably, strengthened my character and mentored me in ways that I am not even yet aware of.

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the US Air Force, Department of Defense or the US Government.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

FIGURE Page

LIST OF TABLES

1. INTRODUCTION TO WORKFLOW BEHAVIOR AUDITING FOR MISSION CENTRIC COLLABORATION

Mission success places high demands on computing. With modern mission-system dependencies [1], process-oriented planning and execution is tightly integrated with computing infrastructure, changing systems management paradigms. Data aware [2] and net-centric [3] axioms of the 1990s and 2000s are quickly being replaced with a mission centric mindset [4][5][6][7]. The old modalities highlighted the ways in which current and future operations would depend on data and networks so as to push computing power to the warfighter. The proliferation of information technology has indeed enabled new capabilities, but the resulting information dependence has not yielded methods to manage networks to support military operations. In contrast, a mission centric paradigm would utilize knowledge of mission planning and execution activities to command and control computing resources and other combat capabilities.

Mission planning and execution requires collaboration among a myriad of inter-organizational entities in a massive scaling and elastic environment. Ranging from DoD, government and foreign partners as well as industry. For example, contractors made up 54% of the DoD workforce in Iraq and Afghanistan over the last decade [8]. At the peak in Iraq, forces were composed of 176,000 troops from 32 nations [9]. In 2008, after Hurricane Ike over 65,000 military, police, fire, and medical responders converged from scores of jurisdictions; manning command posts and dispersing widely over the disaster area [10]. On January 12, 2010, the Haiti Earthquake initiated massive disaster relief

mission for DoD. Within a day, the US military was operating the Port-Au-Prince airport to coordinate relief efforts, and within 10 days, US Southern Command placed 13,657 personnel in the Haiti Joint Operational Area [11].

Despite these complexities, mission centric collaboration requires intense situational awareness to manage the cyber physical dependency. As a spokesman of Department of Defense (DoD) vision, the Secretary of the Air Force summarized information dependency this way, "A great deal of our combat capability operates in cyberspace: command and control systems as well as the intelligence, surveillance, and reconnaissance platforms that ensure battlefield awareness" [12]. With this basis, it is critical that situational awareness seize points of observation on information dependent missions to support high level mission decisions. Operating successfully in this complex mission environment depends on the evidence of mission performance to measure efficient operational planning and effective tactical execution.

1.1. Workflow Behavior Auditing

One opportunity for realizing mission situational awareness is through workflow behavior auditing. Workflow behavior auditing are the functions that "*assess and optimize workflow performance"* [13]. AFRL's aim [14] is ambitious, seeking to understand how every packet that passes through the gateway supports the mission, implying that situational awareness can expose how cyber behaviors contribute to real world missions. In addition to awareness, workflow behavior auditing also has the potential for cost effectiveness for undocumented workflows and legacy systems. For instance, automated workflow behavior auditing can reduce documentation time and

labor costs by factor of 10 [15]. Auditing can also reduce risk in costly modernization programs, such as with the Expeditionary Combat Support System (ECSS). ECSS modernization attempted to consolidate 244 legacy systems, but could not cope with tedious manual auditing and analysis, leading to a $1B failed project [16]. If implemented, workflow behavior auditing can support the larger DoD in modernizing defense business systems, a $6.6B annual effort [17].

Workflow behavior auditing might ensure effectiveness in missions since mission centric collaboration is often designed around workflows. A workflow specifies steps to complete a task in order to drive workflow behaviors. In this study, we are interested in computer based behaviors that are guided by workflows. The behavior must also be observable in the sense that a system or network can be instrumented to identify them, leading to a sequential record of event data, ordered by time and causality.

The event log is the ideal workflow behavior data. At the most basic level, event logs are composed of activities, which are well defined steps in a workflow [2]. In addition to the action performed, an activity may be described by timestamps and people and resources involved in the activity execution. A workflow instance is a sequence of activities and an event log is a collection of workflow instances. Event logs can be used for workflow behavior auditing, where an organization can inspect event logs to assess and optimize mission centric collaboration.

A leading technology in workflow behavior auditing is the relatively young field of (business) process mining [5], which in the last 10 years, has provided tools and techniques for understanding workflow behaviors captured in event logs. Process mining

identifies process aware information systems (PAIS), where the behaviors in the system are driven by workflows [2]. Process mining can discover workflow behavior models, check the conformance of observed behavior to a given workflow model, or enhance a model with situational awareness [18]. Given an observable and repetitive behavior set, process mining is a potential technology to provide situational awareness of mission centric collaboration.

1.2. The Battle Rhythm in Mission Centric Collaboration

Mission centric collaboration conforms to a battle rhythm. A battle rhythm is a routine cycle of command and staff activities intended to synchronize current and future operations [19]. As an example of a battle rhythm, the Air Tasking Order (ATO) is shown if Figure 1.1. The ATO timeline consists of a 72 hour workflow driven planning cycle to task and prepare air units for a 24 hour tactical execution (E DAY) of targeted

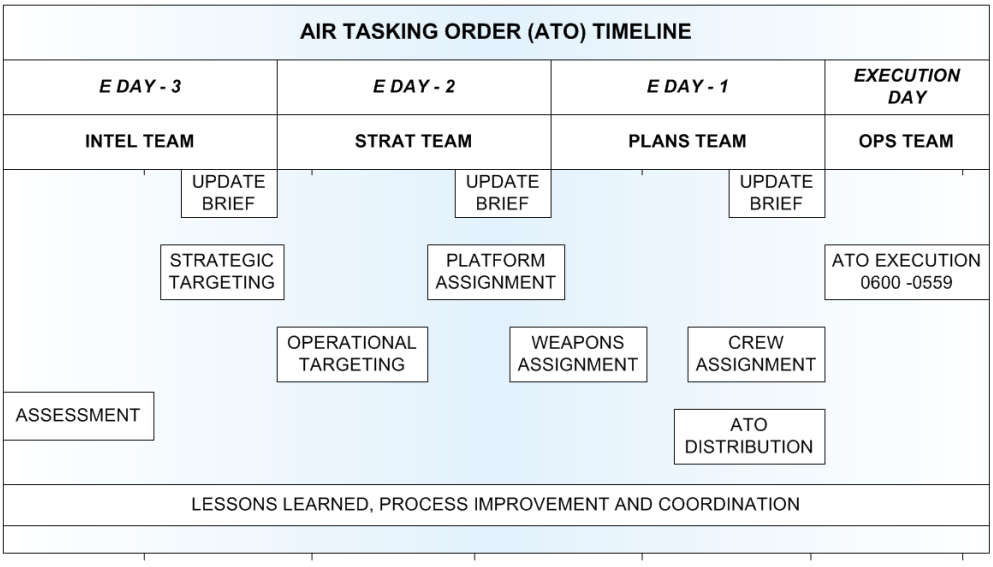

Figure 1.1: Notional Air Tasking Order (ATO) process

air strikes, air to air refueling and air transport. In the first planning day, high level intelligence and strategic objectives identify the air battle. In the second and third planning days, operational and tactical guidance becomes increasingly focused to an end result of assigned crews, flight plans, cargo and munitions [20].

Within this battle rhythm, daily efforts of air operations center personnel and command and control (C2) systems are focused around the ATO process. Workflow behaviors may be observed between persons and systems across many organizations. Changes in mission state and focus occur at discrete times in the process orchestrating the interactions of hundreds of systems interact in a global environment. In this setting, mission centric collaboration generates vast quantities of battle rhythm data. The battle rhythm makes workflow behaviors observable and repetitive, thus a process mining based solution is possible.

Given the battle rhythm, one can envision a process mining based solution for workflow behavior auditing with multiple capabilities (Figure 1.2). First, the auditing framework could detect mission state changes in the battle rhythm from streams of raw event data. Mission state changes could be used for improved workflow models in dynamic settings and accurate assessments of current behavior. Second, the solution could extract workflow instance data from battle rhythm data to create structured workflow event logs. Structured workflow event logs can expand the data sources for process mining tools. Third, the auditing framework would apply modeling techniques to structured workflow event logs. Through process discovery, a battle rhythm analyst could understand person-person, system-system or person-system interactions. Process

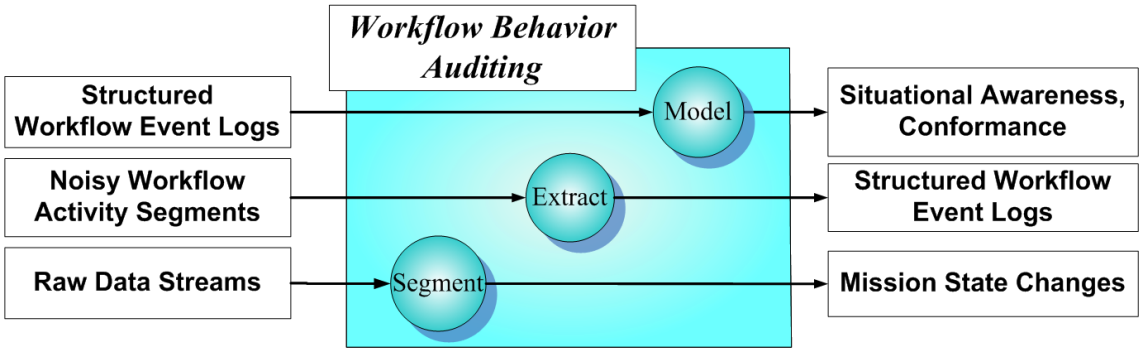

Figure 1.2: Target capabilities for workflow behavior auditing

discovery could also be useful for modernizing legacy systems. Through conformance checking, commanders could ensure conformance of inter-organizational partner and services. A solution that can satisfy the requirements listed above provides a commander with invaluable knowledge to command and control information systems in the battle space.

1.3. Open Challenges in Workflow Behavior Auditing

There are many open challenges to support the battle rhythm in a complex mission centric environment, but three challenges are immediately apparent. The first is finding mission state changes in voluminous data streams of dynamic workflow behavior data. During a daily battle rhythm cycle, workflow behavior may exhibit a dynamic nature as situations, resources and decision making shift the arrangement of activities to accomplish tasks. The battle rhythm is based on DoD doctrine [19] where operational level collaboration meets deadlines for tactical execution. Using C2 systems, collaborators execute workflows to meet intermediate milestones at discrete time intervals. At the boundaries between intervals, workflow behaviors change to meet the

next focus of the battle rhythm, which we refer to as workflow shifts. For real time behavior analytics, it may not be possible to observe systems directly, since they are prioritized for mission critical usage. Thus, passive approaches should be developed, such as identifying mission state changes in network traffic. In this case, the challenge of finding mission state changes is identifying workflow shifts inside packet flows.

The second open challenge is the retrieval of interesting workflow behaviors in exabytes of unstructured information [21]. The problem of auditing systems for situational awareness and conformance checking has become overwhelming for human analysts, but automated auditing solutions require rigid data formats for mining. An unstructured workflow event log can be thought of as a single continuous sequence of activities. To create an event log that adheres to a mining tools data specification, a preprocessing step must search the sequence and aggregate subsequences of interesting behaviors. In mission centric collaboration, the process of automated preprocessing of event logs is made more difficult because of noise from interleaving processes. In short, the workflow retrieval challenge demands a capability to extract workflow event logs for noisy unstructured data segments.

The era of cloud computing, which represents both a business and technological revolution [22], has greatly altered the static and isolated ‘stovepipe’ auditing environment. Now, auditors must recognize the interactions with third party systems and organizations and cope with an elevated pace of change for system baselines [23]. In this era, even data and other resources exhibit an increased level of dynamism. Situationally, resources (or the demand for them) may expand and contract on a moment’s notice to a

massive scale. Since ownership is virtual, resources may be shared outside of the traditional organization, blurring the boundaries of policy. Business processes have exploited the freedom of resource movement, executing transactions a thousand times per second. Computer systems keep the pace, continuing to log activities, but human auditing is inadequate to correlate, reconstruct and evaluate human and system behaviors in a timely manner. However, distributed environments and high speed transactions make it possible that workflow activities will be recorded out of order. For example, distributed environments can exhibit inaccurate clocks that yield inaccurate timestamps. If this occurs, distributed logging devices may record events out of order affecting derived workflow behavior models. Thus, the third open challenge is to enable auditing in cloud services by mitigating the impact of timestamp inaccuracies of a distributed system in a global environment with inter-organizational partners

The open challenges of mission centric collaboration create opportunities for workflow management research. A new era of analytics can make it possible to audit systems for decision support in real time but it will require new technology that can identify mission shifts, extract data from unstructured sources and mitigate timestamp inaccuracy for cloud workflows.

1.4. Research Overview

This dissertation presents research on workflow behavior auditing in the challenges of next generation mission centric collaboration. Decision makers need real-time workflow behavior data to make timely and reliable high level decisions for mission success in a battle rhythm. The mission centric environment is complex, but the

expected conformance to the battle rhythm makes process mining a viable technology for workflow behavior auditing. However, a number of challenges must be addressed before process mining technology can be implemented.

In the next section, it will become apparent that process mining has not adequately addressed the grand challenges of mission centric collaboration. This study will describe the poor performance of process discovery and conformance checking of event logs with dynamic workflow behavior, noise and timestamp distortion. Therefore, we will address a few key near term issues. In sequences of activities parsed from network traffic streams, we identify mission state changes in the workflow shift detection algorithm. In segments of unstructured event logs that contain both noise and relevant workflow data, we extract and rank workflow instances for the process analyst or discovery algorithm. When confronted with timestamp inaccuracy in event logs from semi automated, distributed workflow execution, we develop the flower chain model and discovery algorithm to improve behavioral conformance. For the long term adoption of process mining in mission centric collaboration, we develop and demonstrate an experimental framework for logging uncertainty testing.

In summary, this study will show that it is highly feasible to employ process mining techniques in environments with dynamic mission states and logging uncertainty. By addressing these challenges, we believe that workflow behavior auditing will help decision makers achieve mission success by managing collaborating forces and systems with a mission centric mindset. The impact will be felt in cost effective systems modernization, efficient operational planning and effective tactical execution.

In Section 2, we present the gaps in the current approach of process mining to present our thesis and research approach. In Section 3, we showcase the workflow shift detection algorithm to find shifts in mission state evidenced by statistical changes in activity. In Section 4, we explore log preprocessing techniques to extract and rank meaningful workflow behavior instances. In Section 5, we analyze and overcome temporal uncertainty through the creation of the Flower Chain Petri net and associated discovery algorithm. In Section 6, we detail the design of a testbed and methodology for repeatable testing of process mining methods with logging uncertainty and then conclude in Section 7 with a discussion of future work.

2. PROCESS MINING FOR WORKFLOW BEHAVIOR AUDITING

Workflow behavior auditing can help commanders gain an understanding of mission situational awareness to improve mission management and cost avoidance. However, workflow behavior auditing in the battle rhythm environment is a difficult task owing to the significant challenges of dynamic mission sets, voluminous, unstructured data and globally distributed workflow execution. To utilize process mining for workflow behavior auditing in this environment, we aim to expose the technical issues that currently limit process mining from this application. From a set of near term technical issues, we will develop and execute a research plan to handle dynamic mission states and logging uncertainty.

2.1 Issues in Process Mining for Workflow Behavior Auditing

Process mining is a potential technology to answer the needs of workflow behavior auditing, but there are several technical issues that mining methods can not yet overcome. Situational dynamics affect the activity of worker populations, but methods to detect a mission state change are limited. Behavior data buried in streams of noisy activity logs also limit process mining. Process discovery and conformance checking is mostly applied where timestamp inaccuracies are a non-factor, but in cloud based mission environments, these assumptions are strongly undermined. Though the challenges of applying process mining to workflow behavior auditing are many, we provide an exposition of near term shortfalls that can be immediately addressed.

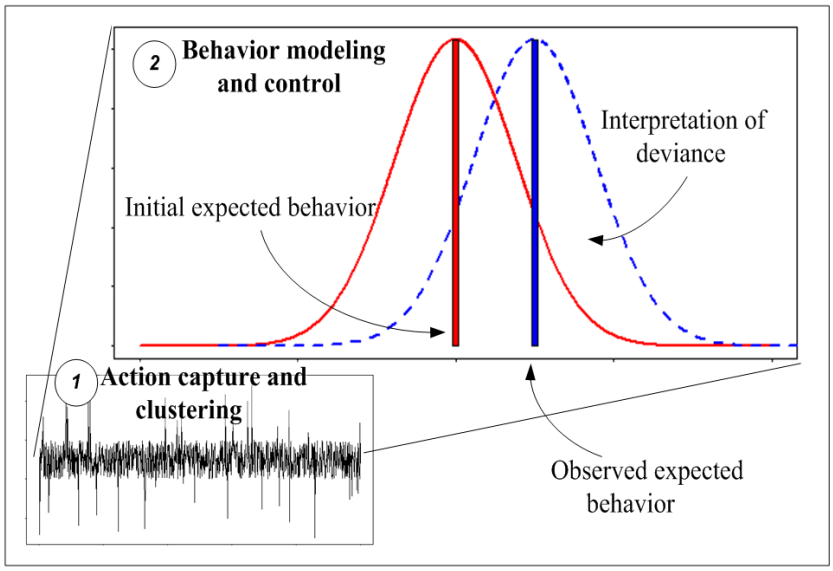

Figure 2.1: Model of expected (red) and observed (blue) workflow behaviors

2.1.1. Dynamic Mission State Identification in Raw Event Data

Workflow behaviors are bound to change rapidly based on the mission state. Deviance (blue line, Figure 2.1) from expected behavior (red line, Figure 2.1) may be detected after an initial model is discovered. In this context, deviance indicates that either the monitored population not conforming to the expectation or that the expectation is wrong. If the expectation is wrong, the new expected behavior must replace the baseline. Assuming that the majority organizational behavior is also the correct behavior, a persistent shift in organizational usage may coincide with a mission state change.

This problem in process mining is known as concept drift [24], overloading the term in the machine learning/data mining community. It has been researched within process mining under the sub field of workflow flexibility [25][26], but the goals of these works are to make the models more flexible to change. Despite these works, most

process mining techniques assume a steady state process [24]. The problem of searching for change points with the log as the only input had not been studied in the process mining community as recent as 2011 [24]. The change point detection approach has been attempted in [26] on workflow management system event logs in a large Dutch hospital [27] to improve segmentation of process models as seen in Figure 2.2. There have been no attempts in process mining to apply change point detection to network traffic to determine changes in mission state. The problem set differs because there is a lot more noise and loss in network traffic as compared to workflow management systems. Mission state changes are common in battle space and disaster response environments and passive network listeners would not inhibit performance of online servers.

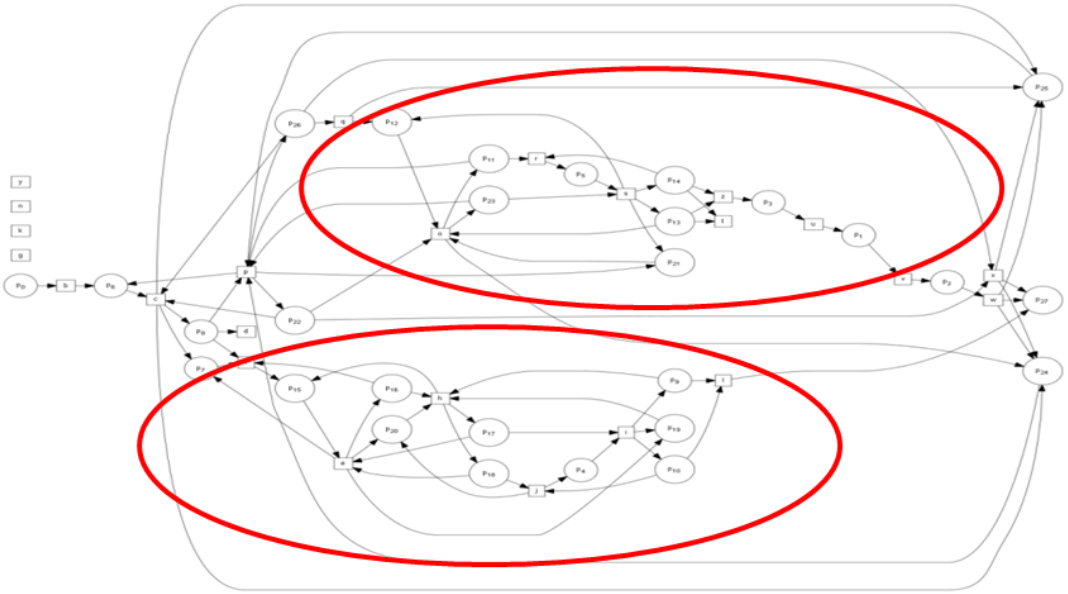

Figure 2.2: Multi-process workflow model due to concept drift

2.1.2 Behavior Extraction from Noisy Event Logs

A vast majority of business data is “unstructured”, meaning that it is not ready to be processed by process mining tools. Raw event logs and packet traces form the basis of training instances to discover behavior models, but current efforts focus on standardization. Standards are materializing as established data formats have emerged and evolved. The first of these was MXML (Mining Extensible Markup Language) [28], but it suffered from being restrictive in the types of information that could be represented and could not be represented in an event log [29]. Still, it is generally easy to use and understand as it follows a simple XML parsing format. MXML was replaced by XES (eXtensible Event Stream) as the standard because of its expressivity and flexibility to capture event data from any domain [29].

By definitions enforced by these standards, process mining data starts with atomic activities to model [2][5]. However, there are few tools [30][31] to extract data from unstructured sources and put them in standard formats. Importing activity event data from network traffic has not been done in process mining. Likewise, few tools have been developed to automate retrieval of workflow behavior from source data to analyst.

Initial works to automate parsing [32][33][34] lack workflow specific ranking criteria to guide automated search. Autonomy is vital to allow the system’s models of expected behavior to change dynamically. In related fields, operations profiles can be extracted from activity sequences [35][36] for the design of mission qualification training programs, but operations profiles have a different structure than workflow behavior. Some parts of the operations profile research is applicable, including profile

extraction of highly repetitive sequences [37] and the use of suffix arrays [38] to gather similar profiles.

2.1.3. Workflow Management in Cloud Computing

Given the wide range of potential inaccuracy in a global distributed system and the movement of business activities to semi-automated and inter-organizational workflows [2], distributed and cloud workflow logging undermines core assumptions in process mining. Service Oriented Architectures [22] pervade cloud technology, interacting at very high speeds in distributed environments. Timestamps may be inaccurate because of clock inaccuracy. The NIST Guide to Computer Security Log Management states the case this way:

"Each host that generates logs typically references its internal clock when setting a timestamp for each log entry. If a host's clock is inaccurate, the timestamps in its logs will also be inaccurate [and] might indicate that event A happened 45 seconds before event B, when event A actually happened two minutes after event B." [39].

A distributed model undermines assumptions of process mining in various ways. Given a log of business activities, the event log can be mined for relationships between activities to form a representation of observed behavior. The α-Algorithm [40] was developed by William van der Aalst, and it stands as the first successful algorithm to model concurrent relationships in Petri nets. We summarize the phases here.

Building Transition Sets**.** Four sets of transitions are inferred from activity sequences in the log: 1)T_a - all transitions mapping the activity language of the

workflow, 2) T_s - start transitions, 3) T_e - end transitions and 4) T_r - repeated transitions, activities which exhibit consecutive repetition.

Rules Inference. The log is searched for four sets of rules. 1) $R_{>L}$ are all occurrences of causation and 2) $R_{\rightarrow L}$ are all occurrences of strong causation, $a \rightarrow_L b$ where $a, b \in A$, b follows a, but a does not follow b. 3) Mixed rules or concurrency relations compose the set $R_{\|L}$ and 4) $R_{\#L}$ and indicates that a never followed b or vice versa.

Place Discernment. Places, P, in the Petri net are determined from the rule sets by separating all $a \in A$ transitions from all $b \in B$ transitions where $a \rightarrow_L$ b. Maximal places are found by removing places whose input and outputs are each subsets of other candidate places. The places are implied states between transitions in a discovered net.

Flow Identification. - Flows, F, are set P to T and T to P using the input and output sets of each place.

The procedure builds a Petri net, $PN = (P, T, F)$ from an event log (Figure 2.3).

Event Log	
sequence	occurrence
abcd	3
acbd	3
aed	2

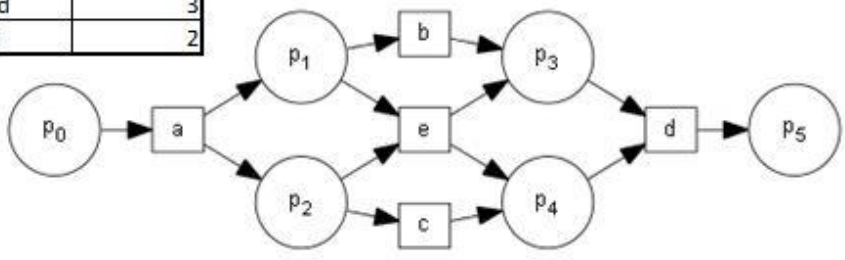

Figure 2.3: Petri net discovered from an event log

There are many approaches in process mining to recreate the workflow model through discovery. These include Hidden Markov Model approaches [41][42][43], directed graphs [44], and transition systems [45]. A great majority of the approaches seek to model Petri nets because workflow management systems use Petri net based modeling languages [2]. Within Petri net algorithms, there are various families: genetic algorithms [46], region based algorithms [45][47][48], heuristics based algorithms [49] and alpha algorithms [40][50][51], which are called alpha algorithms because they extend the first α-algorithm.

While there are many approaches, there is no one good model [2][5][18]. Each process mining endeavor should be driven by an information need to focus the discovery task [5]. For example, a long range goal of this study may be to provide assured and adaptive systems that demands security minded research, and a number of security related approaches exist [52][53][54][55][56][57][58][59][60]. Even so, specific tasks and constraints will be different from system to system. This current work is shaped by limitations in process mining due to timestamp accuracy.

Process discovery events are recorded in a log in order, but distributed execution of activities makes the ordering of events uncertain [61]. Uncertainty from timestamp distortion is detrimental because it can lead to different perspectives on the ordering of activities by different observers. These issues affect the order of events in logs in a decentralized workflow management system, which are critical to process mining. Although attempts to address timestamp inaccuracies exist [62], models that can cope in environments that have timestamp distortion are not common.

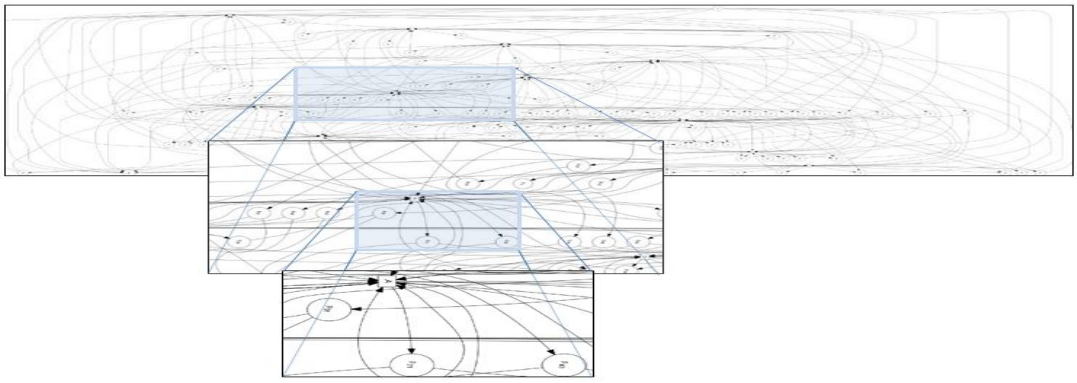

Figure 2.4: State explosion due to concurrency

Timestamp inaccuracy is relatively untouched in process mining but remains a significant inhibiting data problem [63][64]. The result of directly applying process mining algorithms to event logs with causal reordering can be seen in the state explosion in Figure 2.4. State explosion occurs because of an overabundance of perceived concurrency in the model. To handle the timestamp and subsequent state explosion issue for process mining, one must model, analyze and address timestamp distortion. New models must be developed for semi-automated distributed workflows that are tolerant of concurrency but still capable of behavioral conformance.

2.1.4. Logging Uncertainty and Auditing Development

Attempts to directly use process mining tools for mission situational awareness in a dynamic and distributed environment are ill-conceived because the core of research in process mining centers on event logs without system and inference error. *Logging uncertainty* characterizes the possibility of system and inference error in event logs. The Releasability Gateway and RAMP may audit with an incredible amount of logging uncertainty, which may break a fundamental bound of process mining methods.

The issue of logging uncertainty was identified as *"observation noise"* by Rozinat [41]. Some issues pertaining to log quality in general are discussed in [64], yet logging uncertainty is often overlooked and has received no methodological treatment for understanding.

Table 2.1 gives an overview of the type of conditions that may yield logging uncertainty. The issues that process analysts will face in systems with logging uncertainty are varied and numerous. From the body of research that has been discussed, it would appear that process discovery and conformance algorithms can withstand some level of uncertainty, but it is unclear how much. It is also unclear if the auditing technology required in mission centric systems is outside those bounds. Therefore, a testbed and repeatable, working methodology is a critical requirement to applying process mining to environments with logging uncertainty.

Simulation tools for logging uncertainty testing includes Rozinat's HMM based 'Observation Noise Experimenter' [41] as a simulation mechanism to model noise effects that may come from logging devices. Another simulation tool, CPN Tools [65] implements Colored Generalized Stochastic Petri Nets (CGSPN) [66]. CGSPNs add significant value to Petri net simulation in terms of a capability to run parallel process

Table 2.1: Differences of assumptions for process mining environments

Process Mining without Logging Uncertainty	Process Mining with Logging Uncertainty
Centralized logging	Distributed logging
Accurate clocks	Inaccurate clocks
Timestamp synchronization	Timestamp by consensus
Every event is recorded	Event loss is possible
Events are recorded without observation noise	Events may include observation noise
Activity data is atomic [2][5]	Activity data may be inferred from system events

simulation with random timing delays. CPN Tools offers a fast approximation of fairness by selecting enabled markings at random to allow the simulation to explore the state space. Yet to accomplish simulation and observation of logging uncertainty, a more expressive observation framework is required.

A leading modeler and simulator is ProM [67], which is the most prolific research tool used in the field. It is an analysis framework that offers hundreds of plug-ins for analyzing well structured event logs. The framework offers tools for control flow discovery, social network analysis, and historical activity analysis to name a few.

Near and long term issues hinder the adoption of process mining techniques for workflow behavior auditing. With a goal of conforming process mining to the needs of workflow behavior auditing, we will identify a research plan to detect mission state changes, prepare data for process mining and adapt process mining discovery and models for the unique requirements of mission centric collaboration.

2.2. Research Approach

Process mining is not ready for the vision of mission situational awareness, as it is most certainly limited in environments that produce noisy, lossy and time distorted event logs. Process mining performs poorly when analyzing data in noisy, continuous activity sequences. Discovery algorithms have difficulty avoiding state explosion in modeling data that contains timestamp distortion. Extraction and concept drift is difficult even without logging uncertainty, so process mining applications into uncertain environments may not work at all. To proceed towards workflow behavior auditing for mission situational awareness, a number of near term research issues must be addressed.

2.2.1. Thesis Statement and Research Overview

Process mining techniques can be applied to uncertain environments where dynamic mission states and logging uncertainty pollute event logs with noise and timestamp distortion. A repeated methodology of problem analysis and algorithmic development support the claim in order to support auditing mission centric collaboration. Under this thesis, dynamic mission state changes are to be identified, workflow instances must be extracted from noisy, unstructured data and new process discovery algorithms must reduce detrimental effects of concurrency. For research beyond the scope of the dissertation, a testbed must be created to model, analyze and develop process mining algorithms for future issues with logging uncertainty.

2.2.1.1. Workflow Shift Detection in Inferred Activity Streams

The need for detection of mission state changes is illustrated in a battle rhythm scenario where dynamic changes in mission situation yield changes in workflow behaviors. Workflow behaviors, driven by workflow nets, produce expected behaviors that can be observed in event logs. To have situational awareness in a dynamic mission setting, organizations must detect and respond to workflow shifts. In the context of process mining, workflow shifts would be useful in splitting the log between two segments of repeated behaviors. This must lead to process models that are more concise than unsegmented logs.

This study proposes the Workflow Shift Detection Algorithm (WSDA) to segment inferred activity streams into log segments called workflow periods. WSDA will make instance extraction faster and workflow instances that come from them more

concise. The method uses statistical change point detection to identify novelties in streaming data. The algorithm uses generic features of workflow behavior to make the solution general to any streaming log, but is specifically geared to find workflow shifts in Resource Access Actions (RAAs). The algorithm establishes a baseline of RAA features, then proceeds into a sliding window phase with a decision function to find statistically significant changes in an exponentially weighted mean. WSDA is evaluated on its ability to provide more concise process models in the 'Battle Rhythm scenario', a simulated implementation of dynamic workflows in the Convoy Planning System.

2.2.1.2. Extracting and Ranking Workflow Instance Collections

This study proposes the retrieval of Workflow Instance Collections (WIC) through a multi stage algorithm that addresses extraction and ranking of WIC. The data source for the study is unique: a log of inferred activities from network traffic. The control flow perspective in process mining relies on the availability of workflow instances that are reasonably free of noise and expressive of a representative sample of the possible behaviors of a process. The purpose of this study is to overcome the process mining limitation to learn models from noisy, continuous activity sequences. The WIC-Extract algorithm is aimed to find all possible workflow instance candidates, while WIC-Rank offers a set of WIC to a process analyst to support search queries of the activities from an archive. The algorithms are measured by complexity, performance and effectiveness to produce relevant collections to a process analyst. A sample of our resulting workflow models are evidence for using process mining in mission centric computing.

2.2.1.3. Semi-Automated, Distributed Workflow Modeling

After modeling and simulating timestamp distortion, we analyze the core systemic issues that lead to reduced behavioral conformance in process mining control flow models. The issue of concurrency is a direct contributing factor to poor False Positive (FP) rates, but models with inherently low FP rates (Flower Petri nets) are disadvantageous as well. This study proposes a new model that combines positive aspects of both approaches in the Flower Chain Net (FCN) adding detection capabilities for semi-automated and inter-organizational workflows. The FCDA algorithm discovers FCN from an event log. FCN and FCDA are evaluated against existing methods for structural characteristics and behavioral conformance. FCN support mission centric collaboration with a potential method for conformance detection in distributed environments.

2.2.1.4. Logging Uncertainty Testing and Development

The long term vision for process mining in mission centric systems must be supported by a repeatable testing framework to identify problems and develop new solutions. This study addresses this by proposing a testbed and methodology that analyzes the detrimental effects of event logs with system and inference error. We exercise the methodology in this thesis, showing the stages of analysis, problem development and algorithmic solutions. It is a unique tool to process mining, since no other testbed simulates and analyzes timestamp inaccuracy. Thus, a testbed for logging uncertainty testing could be advantageous for the process mining community and our mission situational awareness application.

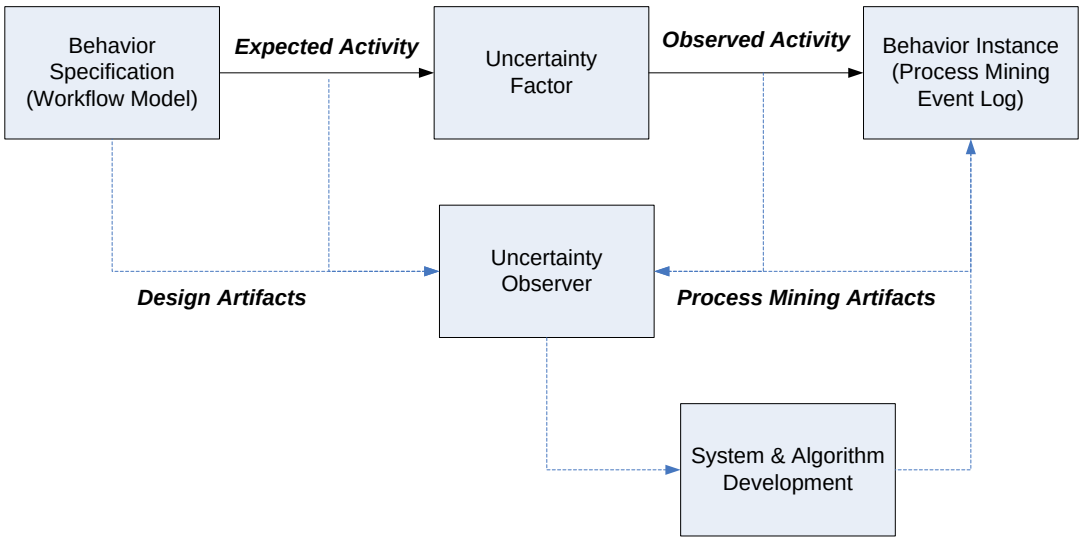

Figure 2.5: Framework for simulating and analyzing logging uncertainty

Such a testbed would be valuable for analysis of uncertainty and devising strategies to overcome its negative effects, but an appropriate observation and control framework is required. Figure 2.5 illustrates our expectation of the solution to be able to observe and control uncertainty factors. Specified from expressive simulation and workflow models, perturbation from simulated logging errors alter the expected sequence of activity to what is actually observed. The uncertainty factor itself is random and unobservable, though dashed lines indicate potential instrumentation points for an process analyst. Having both expected and observed activities, a control-test environment can support a range of experiments with logging uncertainty.

The system will be realized in the Workflow Auditing and Simulation Platform (WASP) and a case study concerned with the degrading structural conformance of models with timestamp distortion analysis. The WASP will be evaluated by comparison to existing tools and methodologies.

2.2.2. *Expected Contributions*

In this thesis, we make the following contributions:

1) We develop a statistical change point detection algorithm that detects concept drift in noisy event logs. The workflow shift detection algorithm (WSDA) uses a sliding window scheme to capture statistical frequencies of general features of resource access actions. We use this to split a log for a process modeling algorithm to find a more concise process model. The algorithm may be used in sequential activity data and not just resource access action streams. We believe it may also be used to reduce extraction requirements for Workflow Instance Collection by splitting logs into workflow periods, where particular behaviors may have a higher rate of occurrence.

2) We develop the Workflow Instance Collection Extract and Rank algorithms to provide a tool for process analysts to extract approximately repeating activity sequences. The two algorithms provide a string processing/information retrieval approach to finding highly repetitive sequences without specific knowledge of the targeted activity symbols. Our workflow behavior features capture inherent characteristics of workflow sequences to improve the ranking of sequences to an analyst. The algorithm also features a genetic algorithm that configures the weights in the ranking function so an analyst can select from top results. We develop a prototype of the RAMP auditor to create models from packet traces. We believe that this tool will be beneficial to the process mining community to expand data sources into network traffic and other unstructured sequential data.

3) We model, analyze timestamp distortion in Petri net models and overcome behavioral conformance degradation through the Flower Chain Net and Flower Chain Discovery Algorithm. Our solution showcases a stepwise scaling approach to allow more concurrent structure into the model than Flower Petri nets. At the same time, it features a minimal footprint while reducing the false positive rate of heuristic algorithms. The solution can be incorporated with anomaly detectors using its structure to carry historical and organizational data. We believe this model could be scaled to incorporate more expressive behavior within a threshold of minimum false positives. We also believe that the model will be useful in managing semi-automated and distributed workflows, where concurrency from timestamp inaccuracy is expected to be high.

4) We design and implement a testbed and repeatable methodology for logging uncertainty testing. The system answers a critical need in the field of process mining to evaluate the robustness of process mining techniques in the presence of timestamp distortion, observation noise and loss. To do this, we propose a simulation framework that observes and transforms activities generated by a colored generalized stochastic petri net in order to emulate the execution of workflows in distributed settings. The methodology is demonstrated throughout our study as empirical evidence of repeatable modeling, analysis and algorithmic design approaches. We believe this tool will be vital to advance process mining technology for mission centric collaboration.

3. WORKFLOW SHIFT DETECTION IN INFERRED ACTIVITY STREAMS*

To apply process mining to workflow behavior auditing, one needs to contend with dynamic mission centric behaviors that conform to a battle rhythm and the problems of inferring activity from raw sources. Inference of human activity from network traffic is possible because packet communication follows a strict protocol. In previous work [68], an activity parser paired with a network sniffing tool harvested data from distributed file system interaction. This data is inherently noisy and even lossy if the inference is imperfect, but it is a suitable challenge for testing the ability of process mining to model mission centric data from unconventional sources.

3.1. Workflows in a Mission Centric Environment

Our main purpose is to infer mission state changes in the observed activities of a computer network. The motivation is to detect cycles in mission centric collaboration from a 'battle rhythm'. A battle rhythm is a routine cycle of command and staff activities intended to synchronize current and future operations [19]. During a daily battle rhythm cycle, workflow behavior may exhibit a dynamic nature as situations, resources and decision making shift the arrangement of activities to accomplish tasks.

A mission centric system can be considered to be a process aware information system (PAIS) if the behaviors in the system are driven by workflows [2]. Within PAIS,

---

*Reprinted with permission from "APSAT: A Framework for Modeling and Analysis of Workflow Dynamics in Mission Centric Systems." by J. Pecarina and J.C. Liu, 2012 in *Proceedings of the 2012 Conference on Collaboration Technologies and Systems*, Westminster, Colorado, 2012, pp. 575–582, Copyright [2012] by IEEE.

workflow modeling languages are used to design workflows and specify human activities to complete a task or objective. Often, the languages map business activity labels to the transitions in a Petri net to identify real world business activities [2]. Workflows designed with Petri net formalism are called Workflow Nets and are used in many workflow modeling products in the industry, including BPMN [69] and YAWL [70]. Workflow designers can verify soundness properties to ensure the flow of activities will eventually result in task completion [5] (i.e., the verification of freedom of deadlock and livelock conditions). Thus the workflow net carries explicit syntax and describes well defined business process activities and workflow behaviors.

If workers behave according to the workflow net syntax, then event logs may capture expected behavior that will conform exactly to the original model. Process discovery algorithms capture expected behavior from event logs by assigning labels to well defined activities and representing them as transitions in a Petri net model [2]. One potential goal for discovery is to recreate the workflow net that defined the original specification of behavior from an event log. This is difficult if a workflow has a changing structure, where an alteration in the workflow net can cause the expected behavior to change. If workers behave according to the new workflow net, then the events in a log that spanned the structural change of the workflow net may consist of multiple sets of workflow behaviors. This is the workflow shift detection problem [68], which has also been called concept drift or workflow flexibility in the process mining literature [24]. In our study, we focus on finding sudden drifts [24], characterized as complete shifts between one workflow and one that follows it. The workflow behaviors

of interest in our problem come from noisy sequences of activities harvested from network traffic. This section illustrates, defines and solves the workflow shift detection problem for inferred activity event streams using an offline and online version of a statistical change point detection algorithm.

3.2. Noisy Inferred Activity Streams

To generate a data set, we devised a battle rhythm scenario involving two scripted workflows (Figure 3.1) from the Convoy Planning System (CPS). CPS is a hypothetical system in a mission centric collaborative environment focused on planning convoy routes in a combat zone. The workflow scripts identify steps to accomplish for subtasks of this mission. Workers executed repeatedly executed a script of activities to accomplish a task (i.e. the intel update task) by interacting with a network file server. In the end, users executed 39 workflow simulations to generate the basis of a suitable data set for understanding dynamic workflows.

CONVOY INTEL UPDATE

You are a convoy watch officer. Please complete the following tasks:

1. CONNECT TO NAS\\PUBLIC
2. GO TO /ConvoyPlanning/SITREPS/
3. OPEN ANY DAILY SITREP IN THIS FOLDER
4. GO TO /ConvoyPlanning/Intel/
5. OPEN THE INTEL UPDATE BRIEF IN THIS FOLDER
6. COPY ANY AMOUNT OF TEXT FROM THE DAILY SITREP AND PASTE IT INTO THE MORNING INTEL UPDATE BRIEF
7. SAVE AND CLOSE THE INTEL UPDATE BRIEF
8. OPEN THE PORTAL HTML PAGE /ConvoyPlanning/Intel/portal.html
9. WHEN IT LOADS, CLICK "NOTIFY LEADERSHIP".

CONVOY CREW READINESS TASK

You are the crew readiness manager. Please complete the following tasks:

1. CONNECT TO NAS\\PUBLIC.
2. GO TO /ConvoyPlanning/Crew/Inbox
3. OPEN A CREW MEMBER RECORD.
4. GO TO /ConvoyPlanning/Crew/
5. OPEN Crew_Roster.xls
6. ADD A LINE IN THE ROSTER FOR THE PERSONAL INFORMATION IN THE CREW RECORD:
   a. ID NUMBER
   b. NAME
   c. TRAINING HOURS
7. SAVE AND CLOSE Crew_Roster.xls
8. GO TO /ConvoyPlanning/Crew/Workflow
9. OPEN workflow_log.txt
10. ADD A LINE IN THE LOG WITH YOUR INITIALS, DATE/TIME, AND FILE PROCESSED.
11. SAVE AND CLOSE THE WORKFLOW LOG.
12. DELETE THE CREW MEMBER RECORD.

Figure 3.1: CPS simulation for battle rhythm scenario

Symbol Table for observed SMB packets			
[0x04,0x00000000]	M	[0x01,0xc0000101]	R
[0x06,0x00000000]	S	[0x72,0x00000000]	A
[0x73,0xc0000016]	C	[0x07,0x00000000]	P
[0x2f,0x00000000]	K	[0x2d,0x00000000]	J
[0x75,0x00000000]	E	[0x32,0x0005,0x00000000]	G
[0x2d,0xc0000035]	W	[0x32,0x0003,0x00000000]	F
[0x2e,0x00000000]	T	[0x32,0x0007,0x00000000]	L
[0x73,0x00000000]	B	[0x32,0x0003,0xc0000148]	I

Resource Action Access Rules for observed SMB Packets

Connect:=A{2,3}BCB{2,3}EE{2,3}F{2,3}
Delete:=G+S{2,3}G{2,3}FIF{2,3}
OpenJpg:=G{2,3}J{2,3}[^L]{1}T+TM+[^K]{1}G{1,2}
OpenTxt:=G{4,5}J{2,3}[^L]*G{2,3}T+M{2,3}[^K]*G+
WriteTxt:=G{2,3}JWJ{2,3}K+L{2,3}M{2,3}S{2,3}P{2,3}G

Figure 3.2: RAA parsing grammar

Activity inference is an intermediate step in detecting workflow behaviors from network traffic. A passive network observer can harvest a stream of bits between users and systems on the network in a packet trace. For this data set, we used Wireshark [71] to sniff and parse SMB [72] packets from bits on the wire and record them as packet sequences. From the packet sequences, we applied a set of rules based on the SMB protocol to describe Resource Access Actions (RAAs). The RAA parsing grammar in Figure 3.2 identified human actions on the file system through regular expression matching. From the 39 simulations we recorded 103,252 packets and inferred 600 activity events from using the RAA parsing grammar. The result of this inference task is seen in the event log in Figure 3.3. A Petri net can be discovered (Figure 3.4) from multiple instances of the executed workflows within the event log.

CONVOY INTEL UPDATE

You are a convoy watch officer. Please complete the following tasks:

1. CONNECT TO NAS\\PUBLIC
2. GO TO /ConvoyPlanning/SITREPS/
3. OPEN ANY DAILY SITREP IN THIS FOLDER
4. GO TO /ConvoyPlanning/Intel/
5. OPEN THE INTEL UPDATE BRIEF IN THIS FOLDER
6. COPY ANY AMOUNT OF TEXT FROM THE DAILY SITREP AND PASTE IT INTO THE MORNING INTEL UPDATE BRIEF
7. SAVE AND CLOSE THE INTEL UPDATE BRIEF
8. OPEN THE PORTAL HTML PAGE /ConvoyPlanning/Intel/portal.html
9. WHEN IT LOADS, CLICK "NOTIFY LEADERSHIP".

Start	End	RAA	Share	Directory
9:29:59.90	9:29:59.92	ATTEMPT	\NAS\PUBLIC	EMPTY
9:30:10.59	9:30:10.62	CONNECT	\NAS\PUBLIC	EMPTY
9:30:11.42	9:30:15.74	TRAVERSE	\NAS\PUBLIC	\\
9:30:16.08	9:30:29.73	TRAVERSE	\NAS\PUBLIC	\Users\
9:30:29.95	9:30:34.42	TRAVERSE	\NAS\PUBLIC	\Users\sim\
9:30:34.64	9:30:37.51	TRAVERSE	\NAS\PUBLIC	\Users\sim\ConvoyPlanning\
9:30:37.52	9:30:50.42	TRAVERSE	\NAS\PUBLIC	\Users\sim\ConvoyPlanning\Intel\
9:30:50.43	9:30:58.97	OPEN	\NAS\PUBLIC	\Users\sim\ConvoyPlanning\Intel\intel_update.ppt
9:31:00.55	9:31:56.41	TRAVERSE	\NAS\PUBLIC	\Users\sim\ConvoyPlanning\Intel\
9:31:56.84	9:32:18.35	TRAVERSE	\NAS\PUBLIC	\Users\sim\ConvoyPlanning\SITREPs\
9:32:18.36	9:32:21.12	OPEN	\NAS\PUBLIC	\Users\sim\ConvoyPlanning\SITREPs\SITREP.pdf
9:32:22.12	9:33:50.37	WRITE	\NAS\PUBLIC	\Users\sim\ConvoyPlanning\Intel\intel_update.ppt
9:33:50.39	9:34:05.16	WRITE	\NAS\PUBLIC	\Users\sim\ConvoyPlanning\Intel\intel_update.ppt
9:34:05.27	9:34:48.24	TRAVERSE	\NAS\PUBLIC	\Users\sim\ConvoyPlanning\Intel\
9:34:48.66	9:34:52.98	OPEN	\NAS\PUBLIC	\Users\sim\ConvoyPlanning\Intel\portal.html
9:34:59.86	9:34:59.97	OPEN	\NAS\PUBLIC	\Users\sim\ConvoyPlanning\Intel\notify.py
9:36:44.33	9:36:44.45	OPEN	\NAS\PUBLIC	\Users\sim\ConvoyPlanning\Intel\notify.py

Figure 3.3: Inferred activity log and comparison to original task script

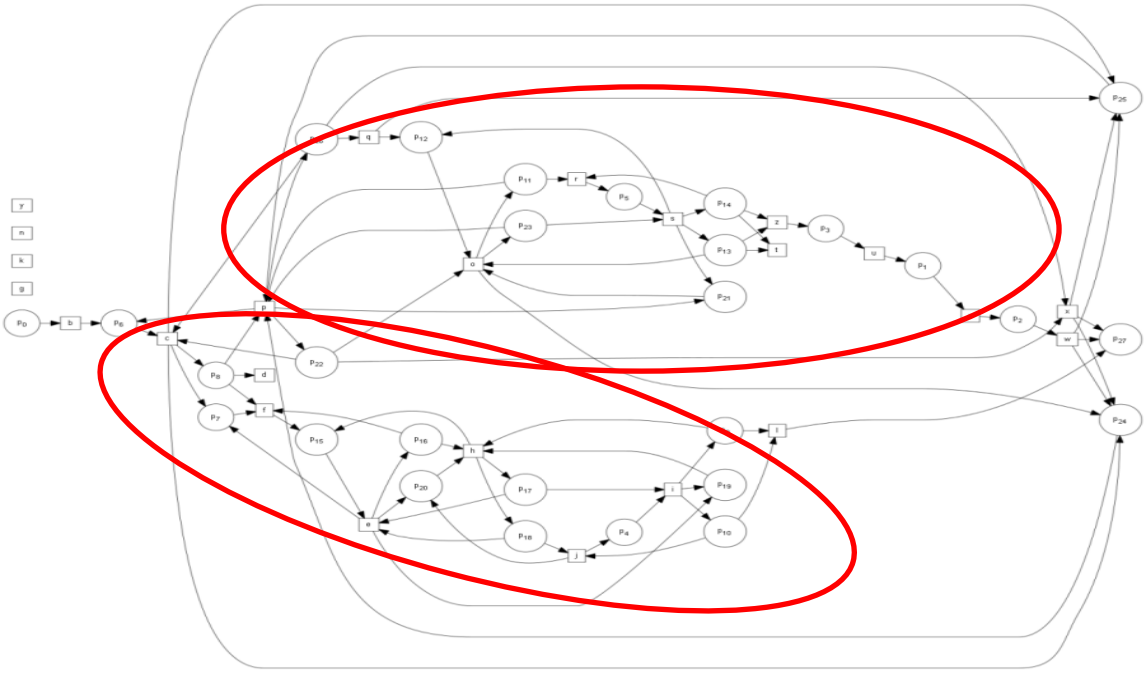

Figure 3.4: Discovered Petri net from unsegmented event log

3.3. Dynamic Workflow Nets

For an intuitive sense of shifting dynamics on a workflow net from this data set, a process mining tool was applied on a manually preprocessed version of this event log (Figure 3.4). The figure outlines two branches of the discovered Petri net, which the mining algorithm assumed to be part of a single workflow reducing the visual understandability of the Petri net. For dynamic workflows, one strategy is to divide the event log before mining the workflow behaviors.

Worker behaviors may follow dynamic workflows, but to detect workflow shifts by observing behaviors one must first be able to define the expected behavior of a worker as specified by a workflow net. It is important to define workflow behavior using the dynamic behaviors of a Petri net in order to ground the perception of behavior to be sequential and repeatable. The formal properties of a workflow net as a type of place transition net (or generally Petri net) are described by van der Aalst [5]. From this we adapt the following definition is from [66] and [5], offering our definition of a workflow behavior instance.

Definition 3.1. (Workflow Net Behavior) - Let $WFNet = (P, T, I^-, I^+, M_0)$ such that:

1. *A **marking** of WFNet is a function* $M : P \rightarrow \mathbb{N}_{\geq 0}$ *where M(p) is the number of tokens in p*
2. $p \in P$ *is **marked** iff* $M(p) > 0$
3. $t \in T$ *is **enabled** iff* $\forall p \in P \mid I^-(p,t) > 0, M(p) > I^-(p,t)$, *denoted* $M >_t$
4. *A **firing** where* $\exists t \in T \mid t$ *is enabled , is the new marking M' of p and p' such that*

$\forall p \in P \mid I^{-}(p,t) > 0 \qquad M'(p) = M(p) - I^{-}(p,t)$

$\forall p' \in P \mid I^{+}(p',t) > 0 \qquad M'(p') = M(p') + I^{-}(p',t)$

Denoted $M_i \xrightarrow{t} M_{i+1}$ *where* $i \rightarrow \mathbb{N}_{\geq 0}$ *indicates the ith firing in the PN*

5. *A* ***firing sequence*** *of WFNet is a finite sequence of transitions* $\sigma = t_m \ldots t_n$, $0 \leq m \leq n$ *such that* $\forall\, i = m \ldots n-1,\ j = i+1,\ if\ t_i, t_j \in \sigma\ then\ \exists (M_i \xrightarrow{t} M_j)$

6. ***Workflow Behavior Instance:***

   (Starting transition) for $t_1 \in \sigma : (\{t_1 \ldots t_n\}\ n \geq 0)\ \exists t_s \in T : I^{-}(p_i, t_s) > 0 \Rightarrow t_1 = t_s$

   (Ending transition) for $t_n \in \sigma : (\{t_1 \ldots t_n\}\ n \geq 0)\ \exists t_e \in T : I^{+}(p_o, t_e) > 0 \Rightarrow t_n = t_e$

   $\forall\, t \in \sigma, \exists t' \in T \mid t = t'$

The definition states that places are *marked* by tokens and arcs from places to transitions are weighted. If the marking (number of tokens) exceed arc weights, then the transition is *enabled*. If the transition is enabled, a *firing* may occur on the transition, yielding new markings for the places leading to and leading from a fired transition *t*. This re-marking is also known as a state transition. When a token are removed from a place due to a firing, the token is *consumed*. When a token is added to a place due to a firing, the token is *produced.* A *firing sequence* is the sequence of transitions that were enabled by firings in the WF-Net. We define a workflow behavior instance, *b*, as an activity sequence from a complete firing sequence of a workflow net starting from an initial transition firing after the input place and ending after firing a transition before the last place in WF-Net. By defining these conditions, the log L_E is defined as the set of all workflow instances expected in a strict workflow environment.

A workflow net may have a dynamic structure, meaning that behavior follows a workflow net that shifts with the passage of time, which we define as a dynamic workflow net.

Definition 3.2. (Dynamic Workflow Net) - A Dynamic Workflow Net is a 6 tuple $PN = (P, T, I^-, I^+, M_0, l, \boldsymbol{\tau})$ where:

1. $\boldsymbol{\tau} = \{\tau_1, \ldots, \tau_k\}$ is a finite and nonempty set of discrete times
2. $P_\tau \in P \mid P_\tau = \{p_1, \ldots, p_n\}$is a finite, nonempty set of places for any time, $\tau \in \boldsymbol{\tau}$
3. $T_\tau \in T \mid T_\tau = \{t_1, \ldots, t_m\}$ is a finite, nonempty set of transitions for any time, $\tau \in \boldsymbol{\tau}$
4. $P \cap T = \emptyset$
5. $F_\tau \in F \mid F_\tau \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs for any time, $\tau \in \boldsymbol{\tau}$
6. $M_0 : P \rightarrow \mathbb{N}_0$ is the initial marking
7. $l : T \rightarrow \Gamma$ is a labeling function of transitions T to a finite set of activity labels, Γ

The dynamic workflow net can be seen as a generalization of the static workflow net, and the original definition is attained if one specifies $\boldsymbol{\tau} = \{ \tau_0 \}$. However, within the dynamic structure, expected workflow behavior is also dynamic such that for any $| \boldsymbol{\tau} | > 1$, multiple sets of expected behavior may be generated by workers. With this context, a dynamic event log can be defined

Definition 3.3. (Dynamic Event Log) – A Dynamic Event Log, L_E , is a set of workflow behavior instances from a dynamic workflow net, such that L_E spans $\boldsymbol{\tau} = \{ \tau_0, \tau_1 \}$

3.4. Workflow Shift Detection

Workflow shift detection is the problem of discovering changes in expected behaviors where a workflow shift may be a result of the introduction of new workflows to a monitored population of users. In monitoring mission centric systems, the event log into workflow periods, $L_{E,\tau i}$ which represent periods of usage that are dominated by a particular workflow. In the initial problem formulation, a log L_E spans $\boldsymbol{\tau} = \{ \tau_0, \tau_1 \}$ and the goal is to split L_E into two $L_{E,\tau 0}$ and $L_{E,\tau 1}$, where τ_1 is the workflow shift in the log.

Definition 3.4. (Offline Workflow Shift Detection). At time τ_0, $workflow_0$ directs worker behavior and at time τ_1, worker behavior is directed by a different workflow. The *Offline Workflow Shift Detection* problem is to find an index $k(\tau_i)$, $\tau_i \in \boldsymbol{\tau}$, $\tau_0 < \tau_1 \mid L_E < k(\tau_1) = L_{E,\tau 0}$ and $L_E \geq k(\tau_1) = L_{E,\tau 1}$.

The finite case can be approached using an offline method hypothesis test [73], but for fast processing, one needs an online algorithm that will alarm at every change in an infinite activity log. Assuming the changes are detected quickly enough and τ is long enough for the detection algorithm to re-baseline workflow behavior, the infinite and online case only requires one shift to be detected [73], thereby allowing for multiple splits of the log. A second change in the definition involves adding the potential for noise in the log. Let L_O be *observed behavior*, $L_O = L_E \times U_N$, where U_N is a stochastic process that inserts activity noise into the sequence of L_E. Therefore, the definition is refined for the infinite case in a noisy log, L_O.

Definition 3.5. (Online Workflow Shift Detection). At time τ_0, worker behavior is directed by $workflow_0$. In observing time ordered sequences of activities in L_O, *online workflow shift detection* is the problem of finding an index $k(\tau_a)$, $\tau_0, \tau_a \in \boldsymbol{\tau}$, $\tau_0 < \tau_a$ such that $L_O < k(\tau_a) = L_{O,\tau 0}$. where the alert time, τ_a, is determined by a stopping rule

$$\tau_a = inf\{k : g(X) \geq h\}$$

With this problem definition, the proposed features will be used to scan the activities in the combined event log to find a statistical change point. Then, WSDA implements a sliding window scheme and slope analysis of local deviations that make up an offline version of the workflow shift detection algorithm. This result is improved with an online version.

3.5. Workflow Shift Detection Algorithm

Having defined the workflow shift detection problem, it is implemented for a stream of logged actions inferred from network traffic. A specific type of actions is very prevalent in file based systems. Without their mission context, these are Resource Access Actions (RAAs) and they are well defined activities since they are matched from regular expressions. Their collection is vital to ascertain ongoing system and mission events for the CPS. In this section, the WSDA detects the split between $L_{O,\tau 0}$ and $L_{O,\tau 1}$ using a mean of feature vectors is a sliding window scheme. This algorithm consists of three phases: 1) baseline establishment, 2) sliding window analysis and 3) significant deviation alert. The section concludes with a proposed online version.

3.5.1. Workflow Features

Workflow shift detection preprocesses the RAA stream to identify measurable features. These features indicate many properties of workflow-driven resource access.

Activity Label - The generic activity label is an indication of the state of the workflow. Different workflows require a distinct relative ratios of activities as seen in Figure 3.5. The Intelligence Update (IU) task on the left has a different mixture or activities than the human resource (HR) task on the right. For example, *"TRAVERSE"* activities are the strong majority in the HR task, while the IU task has twice as many "*OPEN*" activities. The mixture can give an indication of the current workflow.

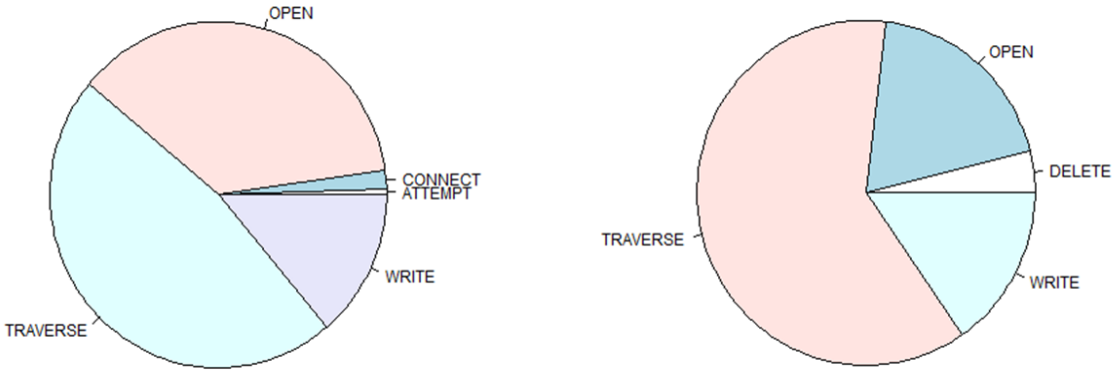

Figure 3.5: Activity labels from two tasks in an RAA event log

Attribute Type - The attributes of an activity may give a hint the target of an activity (i.e. a file type extension, loan amount, etc.). Again, different workflows have a distinct distribution of attribute types. Specific folder names, for instance give an indication of the task.

Actor (Resource) - An actor (or human resource) distribution may be analyzed to see participants involved in the workflow. One may include IPs, User IDs, names of file owners, modification and accessed dates and times if the log provides such things.

Service Delay (SD) - SD expresses the amount of time needed for the system to complete an RAA. Notable delays are due to application, system and network performance and could be an indication of load, constraint or degradations.

$$SD = time\_completed(a_i) - time\_started(a_i)$$

Execution Latency (EL) - EL is the delay between successive RAAs. High execution latency may be caused by many things, including difficult workload, distractions, and poor performance.

$$EL = time\_started(a_i) - time\_completed(a_{i-1})$$

Figure 3.6 shows the changes in mean for SD and EL.

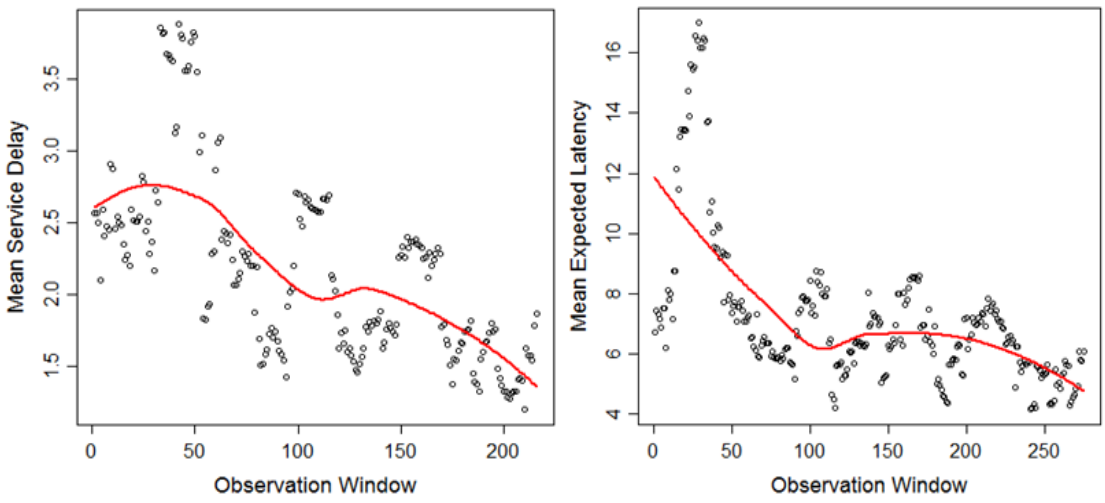

Figure 3.6: Changes in mean service delay and expected latency

To use SD and EL as a feature, we need to discretize the time values by creating a number of bins, defining a bin width and then assigning them. To do so, a normal distribution of service delay times in the log is assumed to be able to divide the observations of *SD(EL)* in the log over quartiles, Q_1 - Q_4, where the bin width is defined by the interquartile range function (IQR) in R [74].

3.5.2. Baseline Establishment

The establishment of a behavioral baseline is critical in workflow shift detection, to identify measurable changes in workflow behavior. The workflow shift detector requires a log analysis tool to identify the relative frequencies of workflow features and then establish a baseline as a point of reference. The algorithm for workflow shift detection will use the workflow baseline to discern novelty. In the initialization step of baseline formation, baselining establishes a feature vector for to hold statistical measurements that correspond to the relative frequency of the presence of a feature within the window. Let m be the size of the feature vector, n be the length of the sequence of activities in L_E and $k = 0 \leq k \leq n$ be an index. Also, let w be a window size for the baseline. Then the initial baseline behavior is a feature vector, over the *long* baseline window, w.

$$\mathrm{X}_L = \{\bar{x}_1 \ldots \bar{x}_m\} \; such \; that \;\; \bar{x}_i = \frac{1}{w}\sum_{k=0}^{k+w} x_i$$

3.5.3. Sliding Windows

After establishing the baseline, a second window (with size c) is initialized to establish the mean of relative frequencies within a *short* change detection window.

$X_S = \{\bar{x}_1 \ldots \bar{x}_m\}\ such\ that\ \bar{x}_i = \frac{1}{c}\sum_{k=0}^{k+c} x_i$ using the same calculation for relative frequencies within the feature sets.

Once the baseline and reference windows have been initialized, the algorithm proceeds through the event log using a sliding index. At each index, k in $0 \leq k \leq n$, an update step is performed for the feature vector X_L (X_S) using a exponentially weighted moving average [73]:

$$X = (1 - \alpha) * \bar{x} + \alpha * x$$

In the update step, the entire feature vector is updated by weighting current events stronger than older ones. The relative frequency of observed types can be seen in Figure 3.7. In the graph, one may observe that recent usage in the short window (last 20 activities) deviates from the long window (last 200 activities) in terms of the distribution of features. The deviations between the baseline (long) and change (short) windows is the absolute value of the difference between the feature vectors.

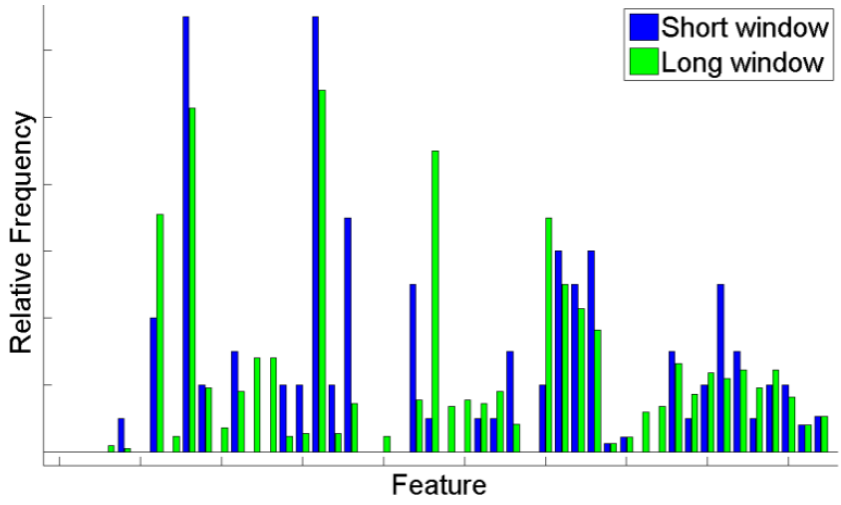

Figure 3.7: Time windowed relative frequencies

3.5.4. Change Point Detection

The significance of these deviations is the object of change point detection. To apply a change point detection algorithm, we need to devise a transformation of the relative frequencies of features into a sequence of scalar time series values. Our approach is to calculate the Kullback-Leibler divergence for an index in L_E. The first step of doing this is to assume that the relative frequencies of the RAA features correspond with the probability of their occurrence. We let P_w be the probability matrix for X_L and P_d be the probability matrix for X_S. The Kullback-Leibler divergence is the sum of the log likelihood for every feature multiplied by the probability of detection of that feature in the short window.

$$y_k = D_{KL}\ (P_d \,||\, P_w) = \sum_i^m \ln\left[\frac{P_d(X_i)}{P_w(X_i)}\right] P_d(X_i)$$

The calculation allows us to establish scalar time series values for the entire event log, $Y = \{y_1 \ldots y_n\}$. The change point detection problem at this point is straight forward, aiming to find the index k where the behavior of the time series varies significantly. As an initial benchmark for change point detection, we implement the Shewhart control chart, and alert by the decision rule

$$\tau_a = inf\,\{k : g(y) \geq h\}$$

where $g(y)$ calculates the change in mean in comparison to the variance in the sliding windows. Therefore, the decision rule is given as

$$\bar{y}(k-d,k) - \bar{y}(k-w,k) \geq \psi \frac{\sigma}{\sqrt{w}}$$

where σ is the standard deviation in the long window and ψ is a threshold for noise.

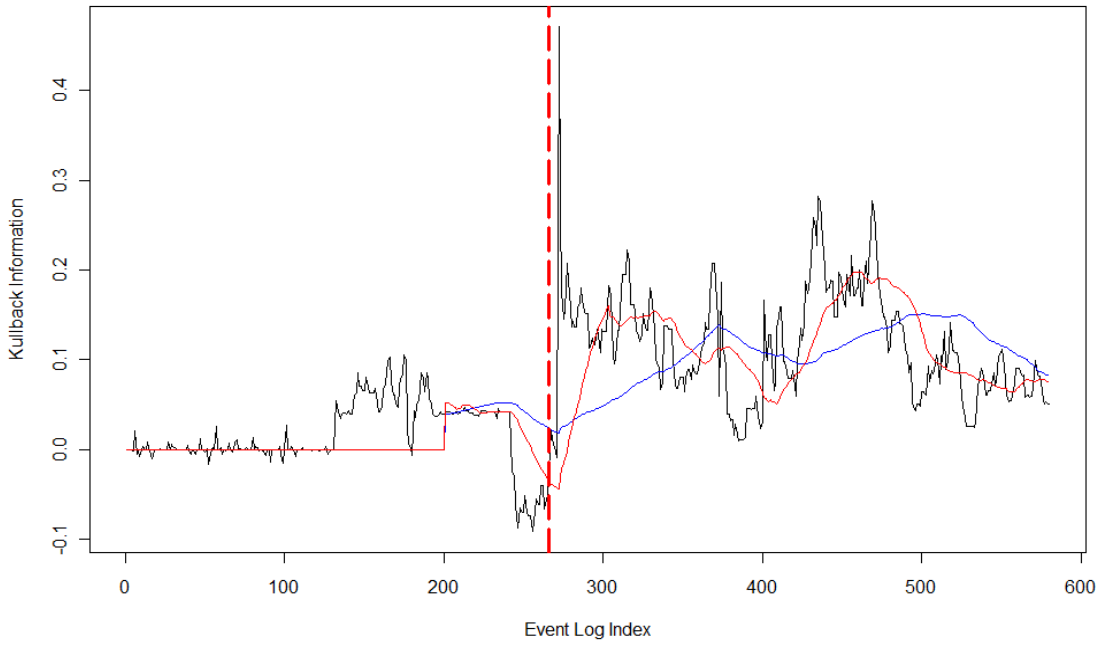

Figure 3.8: Workflow shifts from the CPS simulated workflows

The result of change point detection is seen in Figure 3.8. The solid red line tracks the mean in the short window and the blue line tracks the mean in the long window. The time series, *Y*, is the black line. A change point is detected when the red and blue lines diverge significantly while variance is still small. The dashed red line shows the point in the log where this holds true. Later, change points are not detected when the long and short windows diverge due to the fact that the time series exhibits increased variance. The decision rule interprets these change points as noise.

3.5.5. Online WSDA

The online version of the algorithm requires modifications of two stages of the algorithm. The baseline period and sliding window stages are merged so that the update function is called at every index until a number of observations *w* establish the long

baseline window. The *SD* and *EL* feature bins are created based on the interquartile ranges in the baseline period and shared by the long and short windows. The other modification occurs in the decision function, where the rule becomes

$$\tau_a = inf\{k : g(y) \geq h\}$$

and the slope calculation remains the same. To remain online, the algorithm continues to update and calculate *Y*, but a period of re-baselining must occur. This is acceptable if the time between workflow shifts is greater than the baselining period.

3.6. Evaluation of WSDA

We evaluate WSDA using the data from the battle rhythm scenario in CPS. WSDA will detect a change point in the log, but the correct change point will yield a more concise model. After evaluating WSDA based on this measure of effectiveness, we review the complexity of the algorithm.

3.6.1. Effectiveness

WSDA was applied to the activity event stream in the battle rhythm scenario for the Convoy Planning System. The workflow shift that was detected by the algorithm selected a time instance after two executions of the second workflow and the completion of the first. Using this time instance, the log files were split in two. Workflow instances in the logs were manually extracted to create the Petri net models in Figure 3.9 and Figure 3.10 using a heuristic modeling algorithm [49]. The heuristic modeling algorithm identified activities that were under the threshold of occurrence, which would be seen as nodes without arcs in the figures. The effectiveness will be measured by the ability of the workflow shift detection algorithm to create more concise workflow models.

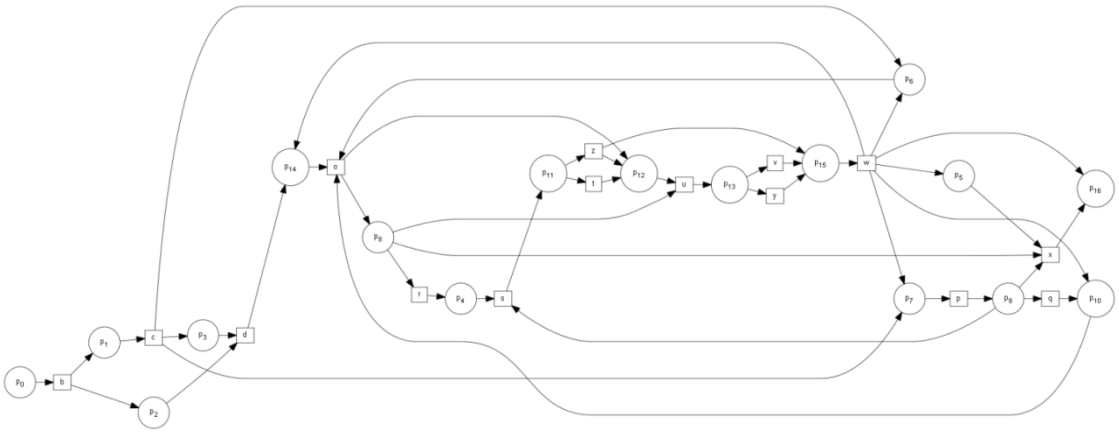

Figure 3.9: Workflow 1 in the battle rhythm scenario

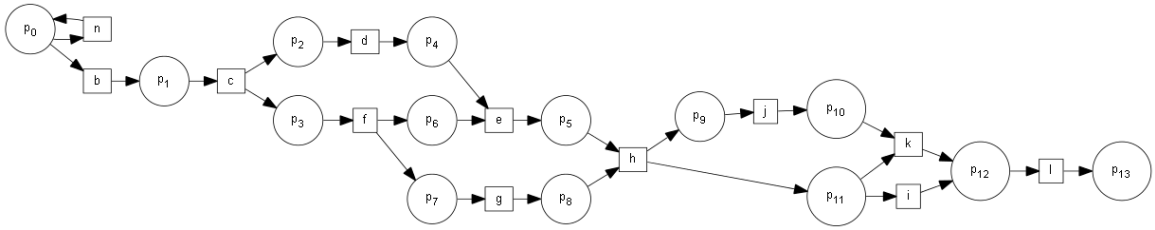

Figure 3.10: Workflow 1 in the battle rhythm scenario

We determine the effectiveness of workflow shift detection to split the log into logs that concisely represent behaviors. The heuristic miner created the Petri net in Figure 3.11. In the figure, there are two branches of the discovered Petri net, which the mining algorithm modeled as a single workflow. The boxed area shows a region of the workflow that possesses a highly similar structural conformance. The footprint conformance metric [5] is 0.987 for the activities within the box in Figure 3.11 and all the activities in Figure 3.10. The result suggests that segmentation of the log file can lead to exposing a concise substructure.

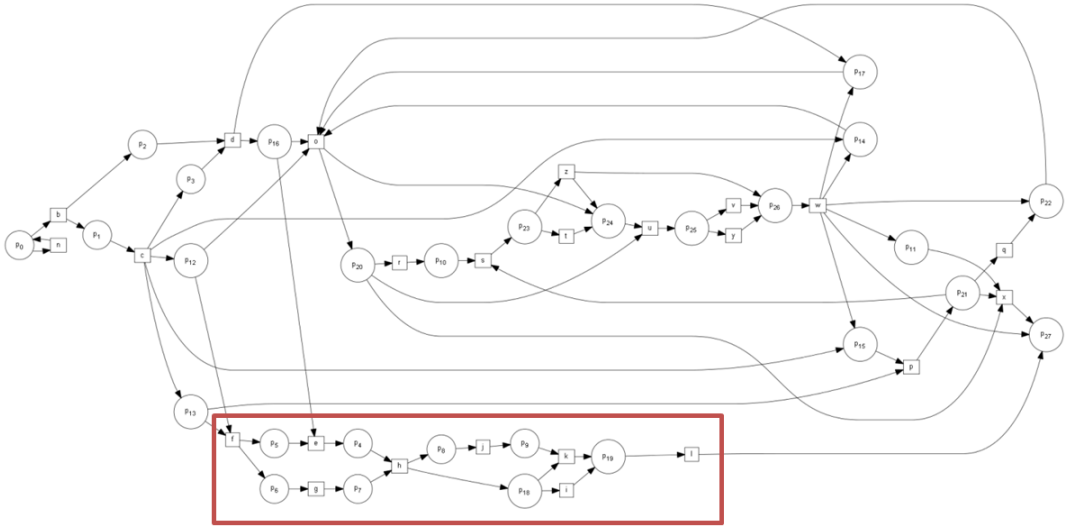

Figure 3.11: Petri net from unsegmented log with highlighted workflow

3.6.2. Complexity

The analysis of WSDA with regards to complexity is straightforward. The WSD Algorithm is $O(wm)$ for storage, storing only scalar values of CFD and the vector of sliding window means (m) over the window size w. For runtime complexity, baseline establishment is $O(wm + cm)$ where wm is the dominating term. Baseline establishment is the bottleneck for performance in the system, since the sliding window scheme uses an $O(1)$ update step. In total, the algorithm has a runtime complexity $O(nw)$.

3.7. Discussion

WSDA can be an effective tool for segmenting log files for process analysis because the segments can assist the process analyst in choosing a point in the log that will provide concise process models. Concise models are important for conformance checking tools to focus on a set of behaviors that are relevant to the situation. Models

that are provided through the use of this tool may give situational awareness tools a means to report and adapt to mission states.

However, the process of creating these models required manual extraction, so WSDA has limited potential for producing better workflow models on its own, since it requires a human analyst to extract behaviors from the resulting logs. In the context of adaptation, the ability to segment logs is incomplete without an accompanying tool to extract the workflow instances from a raw event log. This inadequacy is addressed by extraction and ranking of workflow instance collections.

4. EXTRACTING AND RANKING WORKFLOW INSTANCE COLLECTIONS*

Mission centric collaboration generates raw event data which need to be preprocessed before auditing. In process mining, the source of knowledge for the auditing task is the activity event log. An activity event log is composed of well defined steps in a business process (activity) arranged sequentially and recorded in collections (event log). However, workflow behavior data from mission centric collaboration may be streaming in bits specified by communication protocols rather than in activity event logs. The task of preprocessing this data stream to create an event log occurs in two stages, activity inference and workflow instance extraction.

4.1. Process Mining Data and Discovery

The process mining consortium has established event log formats to standardize process mining efforts [28][29]. The well structured formats separate finite activity sequences that are called workflow behavior instances. One could think of the event log of a collection of sequences, where the sequences in these collections may have some similarity to each other. This will be advantageous since an expressive behavior model can be represented by a collection of workflow instances that bear similar patterns to other instances in the collection. This is illustrated in the event log in Figure 4.1a, where

*Part of the data reported in this section is reprinted with permission from "Behavior Instance Extraction for Risk Aware Control in Mission Centric Systems." by J. Pecarina and J.C. Liu, 2013 in *Proceedings of the 3rd International Conference on Cognitive Methods in Situation Awareness and Decision Support*, San Diego, California, 2013, pp. 45-50, Copyright [2013] by IEEE.

Event Log	
sequence	occurrence
abcefg	3
acbefg	3
adefg	3
adefgefg	2
abcefg	3
acbefgefgefgefg	1

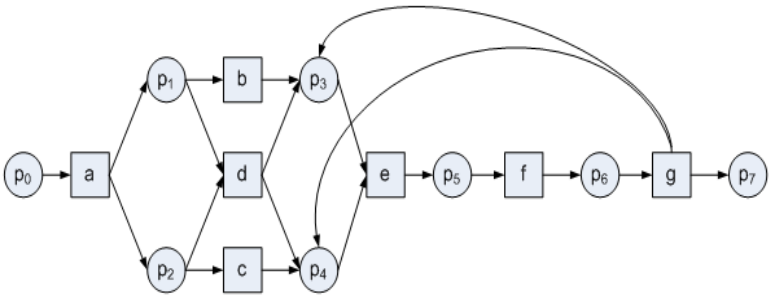

Figure 4.1: Event log (a, left) and corresponding process model (b, right)

similarity is apparent by the repeated use of certain symbols, first and last symbols and subsequences.

Further elaboration of the properties of target workflow sequences can be illustrated using the Workflow Net (WF-Net) in Figure 4.1b. First, one can identify invariant paths of the WF-Net where all workflow executions in the event log pass through (*'efg'* in the figure). The invariant path is executed exactly in all samples in the workflow log. Second, there may be looping substructures, (the repetition of *'efg'*). Third, there is the potential for variance between executions. The concurrent activities *'b'* and *'c'* both must execute when either is executed, but order may be transposed. Also, the activity *'d'* may be substituted for *'bc'* and *'cb'* through insertion and deletion. As shown in Figure 4.1a, the collection of these sequences is an event log and the process model in Figure 4.1b can derived from the log using a process discovery algorithm.

Our goal is to find collections of workflow instances that will be useful to a process discovery algorithm. The focus is on segments of inferred human activity from packet traces that have been segmented by a workflow shift detector. Because of

inference and faulty observation, the activity sequence may be lossy or contain noisy activities. A direct application of process mining to this log will yield poor results, but by extracting workflow instance collections and presenting ranked collections to a process analyst, the task of preprocessing can be automated to the point of selection. Our technical contribution is an algorithm that automates extraction and ranking of workflow instance collections without *a priori* knowledge of activities in observed workflow behaviors.

4.2. Event Logs from Inferred Activity Sequences

In the context of process mining, a workflow instance collection can be considered an event log of filtered and structured process mining source data. As opposed to this, raw and unstructured source data are finite segments of an infinite log that contain workflow instances, but may be surrounded by noisy activities from other processes. A progression from process mining event logs to contiguous activity segment logs with increasing levels of noise illustrates the problem. The target log for workflow instance collection extraction and retrieval is inferred from packet data where extraneous activities may pad the gaps between workflow instances of interest. In the next section, workflow instances and the problems of extraction and retrieval are explicitly defined.

4.2.1. Continuous Sequence Logs

Before defining the problems, one must analyze the effects of noise and continuous sequencing on process mining algorithms. The simple example of an event log, L_1, and discovered *WF-Net* is in Figure 4.2; observing that causal relationships are represented as flows across places. In other words, the observation of *'ab'* in the first

sequence is perceived to be a causal relationship, *a* leads to *b*. Concurrent relationships are inferred when events exhibit transposed order (i.e. *'bc'* and *'cb'* in the first two instances). The reader should refer to [5] for a thorough exposition of process discovery, but the importance of the figure is to note how clean and structured data produces a concise process model.

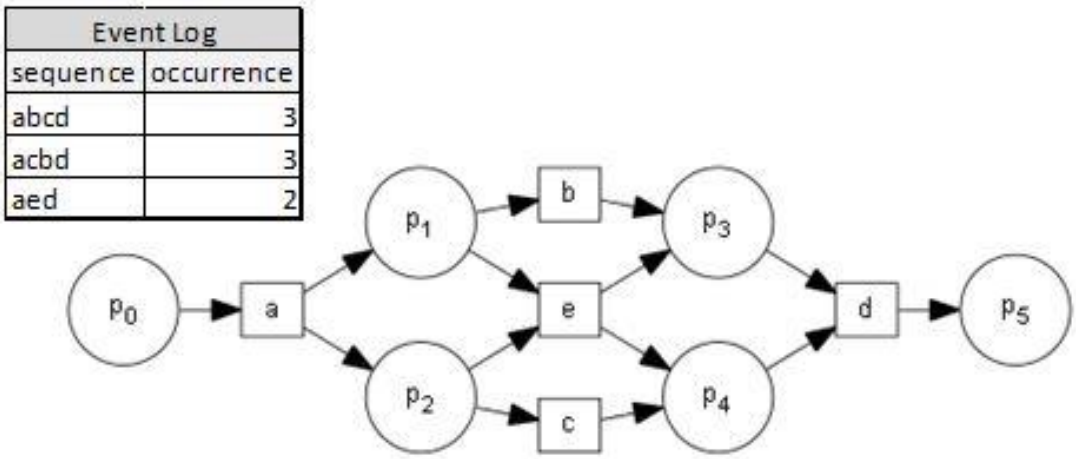

Figure 4.2: Simple event log and Petri net exhibiting concurrency

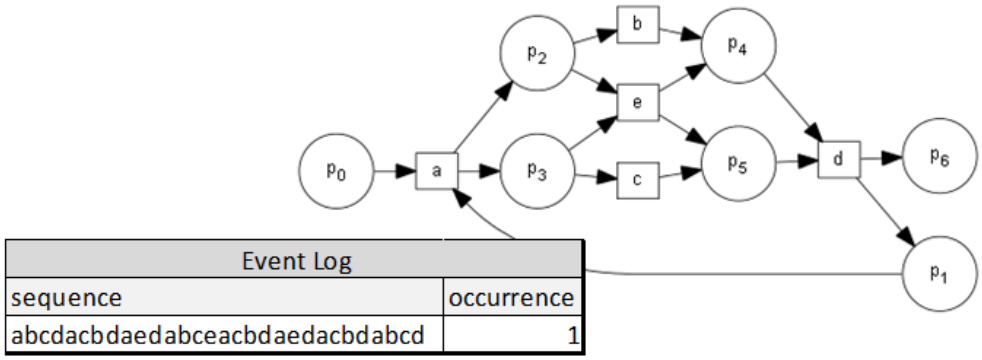

Figure 4.3: Continuous sequence event log and Petri net with circular flow

On the other hand, when workflow instances are in unstructured sequential log files, they tend to look like Figure 4.3. The process mining algorithm was still able to discover the log from L_2, but the graph includes a circular flow from the last transition to the first. This is not a significant issue, since the graph is otherwise very similar visually and in terms of the conformance of the underlying transition rules. The footprint conformance of the rules for this Petri net is $1 - \frac{different\ rules}{possible\ rules} = 1 - \frac{2}{25} = 0.92$. Yet the stage is set for increased perturbation of the activity event relationships.

4.2.2. Noisy, Continuous Sequence Logs

Figure 4.4 shows how noise in the continuous sequence log increases ambiguity in a process model. When other activities occur before or after the workflow instance and before the next, a log may exist where instances are padded with noise (L_{3a}) and are juxtaposed (L_{3b}). A heuristic method [49] can reduce noise, yet the structure is increasingly compromised (footprint conformance = 0.67).

Event Log	
sequence	occurrence
abcd$	1
#acbd	1
aed(*	1
abcd{	1
acbd	2
aed	1

Event Log	
sequence	occurrence
abcd$#acbdaed(*abcd{acbdaed	1

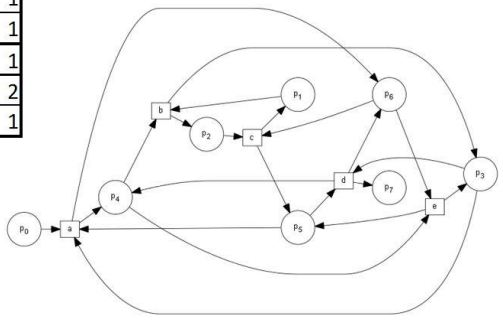

Figure 4.4: Noisy, continuous event log and discovered Petri net

However, the heuristic method has limitations with increased noise in a noisy, continuous log. Increasing noise padding from other processes in the log file in simulation, the Petri net looks progressively worse (Figure 4.5), as noisy activities are added to the Petri net until the original process is indiscernible.

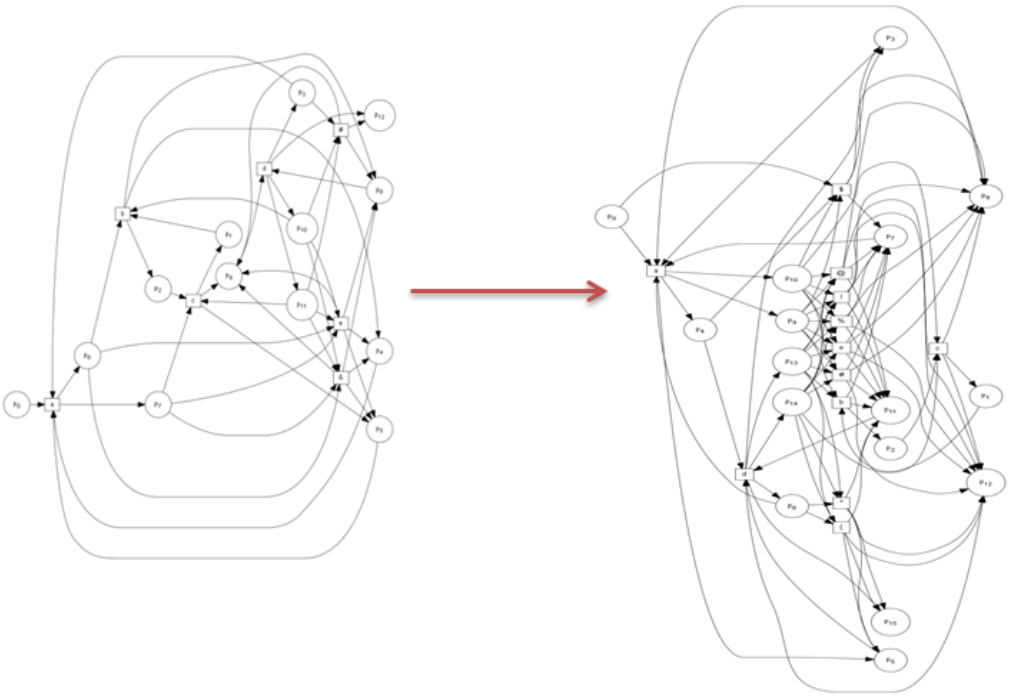

Figure 4.5: Exploding process models with increasing levels of noise padding

The heuristic method reduces noise when the noisy activities occur within the workflow instance, but noise on the edges of the activity sequences do not get reduced using the method. WSDA segmented the activity stream, but it left another problem in extracting and retrieving the workflow instances from a noisy, continuous sequence log file.

4.3. Workflow Instance Collection Extraction and Retrieval

Our aim is to find workflow instance collections for process mining from inferred activities, therefore this section describes the Workflow Instance Collection Extraction problem from a sequence S of labeled activities $a \in A$. A workflow instance collection is a set W composed of similar candidate workflow instances with at least one stemmed candidate exactly matches another stemmed candidate in the set. For clarity, the composition of candidate workflow instances and the nature of stemmed candidates are defined first.

Definition 4.1. (Candidate Workflow Instance) - *A Candidate Workflow Instance (CWI) is a sequence of labeled activities* $a \in A$ *such that* $CWI = a_p, a_{p+1}, \ldots a_q$ *where* $0 \leq p \leq q$. *It is denoted CWI[p,q] to indicate its indices within S.*

The candidate workflow instance is a subsequence of S, but looping substructures (as seen in Figure 4.1) allows workflow execution to repeat portions of the workflow specification. We refer to this as a CWI Loop.

Definition 4.2. (CWI Loop) - *A CWI Loop (CWIL) is a sequence of labeled activities* $a \in A$ *within a CWI such that*

1) $k = i - j$ *is a non-negative, non-zero length of the CWIL*
2) $CWIL = \{a_i, \ldots a_j\}$, $p + k \leq i < j \leq q$ *is a substring of S and CWI*
3) *For* $CWI[m, n]$ *where* $n - m = k$ *and* $n = i - 1$, $CWI[m,n] = CWIL$

That is to say that *CWIL* is the second part of a tandem repeat within the *CWI*. This is an important distinction because *CWIL*s must be stemmed to make it possible to identify like candidates. The *WF-Net* gives no limit on loop size, so by removing them, it is much more likely that a string distance function will find them to be similar. The similarity we assign is the *ε-similar CWI property* based on the edit distance [75] function, denoted *ED*(,).

Definition 4.3. (ε-similar CWI Property) *Given sequences, g and g', ε-similar CWI property is satisfied when ED(g, g') ≤ h . h is a threshold defined by h = length(S) * ε where 0 ≤ ε ≤ 1.*

The definitions of the *CWIL* and the *ε-similar CWI property* allow a relationship between *CWI* in *S* to become clear.

Definition 4.4. (Candidate Workflow Instance Similarity) - *CWI-Similarity is a comparison function $S(CWI_1, CWI_2)$ performing the steps*

1) *Stem(CWI) is a stemming function | ∀ CWIL ∈ CWI, CWI - CWIL*
2) *CWI* = Stem(CWI)*
3) *$S(CWI_1, CWI_2) \rightarrow \mathbb{N} = ED(CWI_1^*, CWI_2^*)$ where ED(,) is the edit distance function*

CWI-Similarity depends on the stemming function to relate workflow instances to each other. Under this framework, it is possible to group similar instances in a collection.

Definition 4.5. (Workflow Instance Collection) - *A Workflow Instance Collection (WIC) is a set of CWI with the following properties*

1) $\forall i,j \in |WIC|, S(CWI_i, CWI_j) \leq h$
2) WIC_h *is the epsilon similarity for all candidate workflow instances in WIC*
3) $|WIC| > 2$
4) *Exists* $i,j \in |WIC|$ *such that* $S(CWI_i, CWI_j) = 0$ *and* CWI_i *is the centroid of the set*

Thus the *WIC* consists of sequences of activities that hold the properties of workflow instance collections and are epsilon similar to each other. The requirement that at least one *CWI* in the collection exactly matches another in the set is both practical and desirable. Practical because it limits the amount of sequences that must be checked to find a workflow instance collection and desirable because workflow discovery and conformance algorithms often use heuristics based on exact repetition in the log.

The definition of a *WIC* is framed in a way to address two problems. The first is to identify all possible subsequences with the characteristics of a *WIC*. However, in a noisy activity log, the ability to find these collections is not sufficient without defining domain specific workflow information. Since our formulation is open ended, a second

problem is apparent, selecting the right *WIC* for the process analyst. The first problem is WIC Extraction and the second is WIC Ranking.

Definition 4.6. (Workflow Instance Collection Extraction Problem) *The Workflow Instance Collection Extraction Problem (WICEP) states that given a sequence S, find all sets of* WIC_h *for any given h.*

All sets require considerable enumeration (worst case ($O(2^n)$) of the sequence *S*. With such a large search space, ranking and heuristics will be necessary. That fact gives credence to the need for a ranking function for the extracted instances.

Definition 4.7. (Workflow Instance Collection Ranking Problem) *The Workflow Instance Collection Ranking Problem (WICRP) states that given a sequence S, rank all sets of* WIC_h *for any given h according to a set of criteria C and ordered using a weighted ranking function R(C,WIC)*

$$R(C, WIC) = \Sigma\, w_i * c_i(WIC)$$

where $c_i(WIC)$ are weighted functions in *C* that evaluate the *WIC*. Thus *C* has the capacity to evaluate *WIC* for its 'interestingness' to a process analyst. The problems are separated to yield different solutions for extraction and ranking and to provide clarity, but in implementation they can be intertwined to save processing time and introduce heuristics.

4.4. WIC-Extract

A solution for WICEP is the WIC-Extract algorithm. WIC-Extract accepts an input sequence *S* as input and returns all possible WIC to the WIC-Rank algorithm. First, *repetition stemming* prepares *S* for approximate repetition analysis. In the second stage, *grapheme analysis* populates a *workflow suffix trie, G* while determining the prevalence of behavior in *S*. In the third stage, *ε-similar aggregation* populates *WIC*.

4.4.1. Repetition Stemming

Consecutive repetition in workflow behavior occurs in event logs if failed system interactions (e.g. unsuccessful logins, incomplete forms, re-accomplished steps, etc.) force repeated actions, cycles or loops. Rather than trying to handle this issue by expanding *ε*-similarity, which may introduce noise in the instances, the *S* is pre-processed to remove looping structures, remembering their indices to reintroduce during the modeling phase.

The algorithm for repetition stemming (Figure 4.6) involves multiple passes of the *S* to remove loops starting with the smallest and ending with the largest. Otherwise, attempting to remove sequence loops in any other order causes some repetitions to be missed. Consider as an example the phrase HELLOHELO. In the first pass, the L is removed to produce HELOHELO. Without this, when *step* is 4, it will not detect the loop. Three indices track the locations of consecutive strings in *S*. To advance the indices, the *hammingDist*( ) function checks the equivalency of symbols using Hamming distance and returns the number of misses. Hamming distance is preferred over edit distance because we seek an exact match and it is faster since it assumes sequence

alignment. After the check, the next exact match cannot in *S* occur until the indices iterate past these misses.

```
Repetition_Stemming(S* = S, maxRep):
    step = 1        // Loop Size
    while step < maxRep :
        miss = 0
        idx_l  = 0
        idx_m = idx_l + step
        idx_r = idx_m + step
        while (idx_r) <= |S*|:
            str_1 ← S* [idx_l , idx_m]
            str_2 ← S* [idx_m , idx_r]
            miss = hammingDist(str_1, str_2)
            if (miss == 0)
                    // Store the match and its position
                    S* ← S* [0 , idx_l] + S*[idx_r , |S*|]
             // Increment by miss (rechecks this idx)
             idx_l, idx_m, idx_r += miss
        step = step + 1        // Repeat for next loop size
    return S*   // Stemmed
```

Figure 4.6: Repetition stemming algorithm

4.4.2. Grapheme Analysis and Workflow Suffix Trie

In the second stage of *WIC-Extract*, n-grams are expanded out of the repetition stemmed *S* into sets in *G*. In most settings, n-gram expansion concentrates on particular values of *n*. In the workflow modeling domain, behavior instances that capture large portions of workflows are needed, so *WIC-Extract* does not necessarily restrain the value of *n*. In addition, later stages in the algorithm make use of activity frequencies. Thus, the algorithm starts with n-grams of 1 and searches for a *maximum N (Max-N)* in the input stream. *Max-N* is equivalent to the length of the longest repeating substring in *S*. Finding Max-N is done at the same time as repetition stemming, using a suffix trie to store the

matches of the repeats for future approximate matching and ranking steps. *Max-N* and *Min-N* are also available as user defined parameters, leaving the final determination of *n* with the user who has intuition about his particular type of workflows of interest.

The key point in grapheme analysis is the data structure used to contain n-grams $G(n)$. While the set of all n-grams G may be maintained in a database, visualizing a set of n-grams $G(n)$ using a suffix trie is valuable to understand how activities are grouped before extraction. A trie is a tree based data structure that stores a null at the root and each node stores a symbol, except at the leaf nodes which holds the index of the n-gram.

An example trie in Figure 4.7 shows partial construction of the trie using the reference string. Construction (Figure 4.7 (1)) for a string *abcefgacbe* is shown where the first suffix (the entire string) has been inserted to the tree along the left side of the figure. The second insertion into the suffix trie, *G*, is *bcefgacbe*, leaves the *a* out of the string, starting a new path from the root. Each string and substring in *S** is added in the same fashion.

At the same time, the suffixes of S* are also scored in *G*. As an example, *'efga'* is added and the final node records term frequency, gram criteria, index, a repeating bit and other workflow specific criteria that can guide search. After creation of the suffix trie, the repetitions that were removed in repetition stemming are also added to G.

The result of this step is creation of a data structure that will make search faster. To find whether the string *efg* was ever repeated, one merely needs to traverse the trie by parsing *'efg'* and inspecting the node for the number of exact matches that were within the original string S, repetition included.

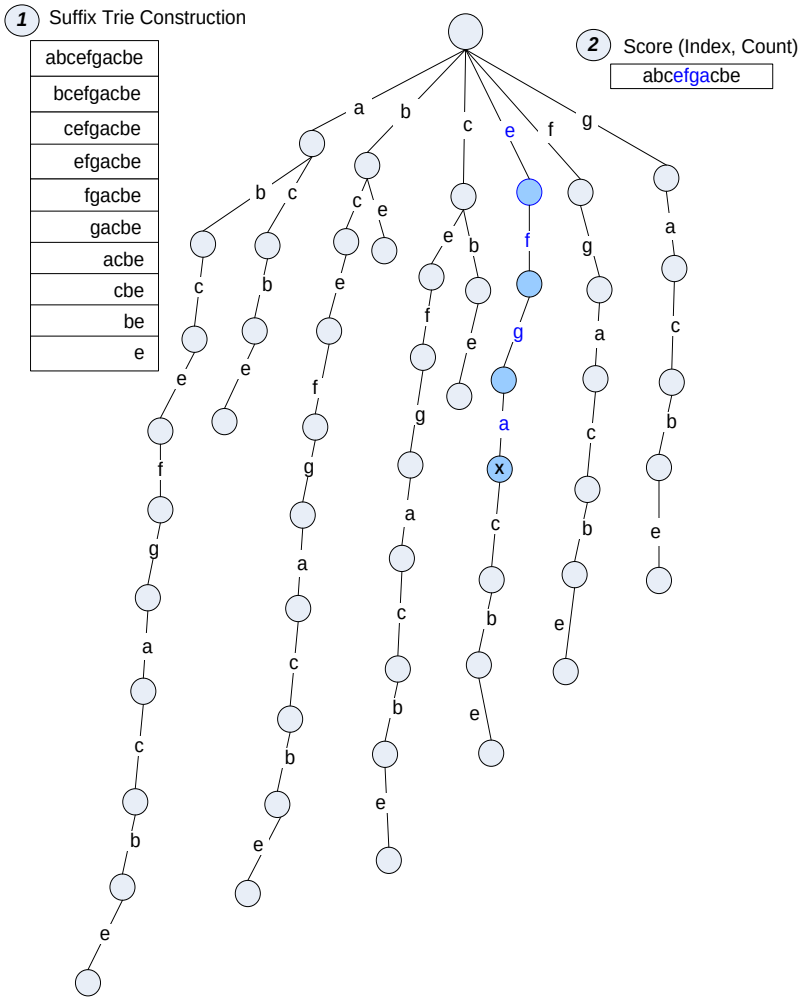

Figure 4.7: Workflow scored suffix trie

4.4.3. ε-similar Aggregation

With all n-grams in the suffix trie data structure, the algorithm proceeds to ε-similar aggregation. The basic premise is that grams are selected from the trie from longest exact matches to the smallest. The workflow suffix trie stores useful information to guide the search, first in the presence of exact matches to serve as seeds for *WIC*s, but also in the indices, so overlapping strings are not checked.

```
ε-Similar_Aggregation ( G , ε-dist, M ):
   List allWIC                  //initialize WIC list
   for (m = M ; m > minM; m--)
      for (out = 0 ; out < |Gm|; out--)
         g_out = Gm ( out )
         CWI c = g_out          // create a seed CWI from g_out
         WIC.add(c)             // start a new variant data structure
         for (e = m- ε-dist ; e <= m+ ε-dist ; e++)
            // need to check variants in ε-dist tries above and below n
            for (in = 0 ; in < |Gm|; in--)
               g_in = Gm ( in )
               if (abs(g_in.index - g_out.index) >= m)
                  // continue check only if g_out is not overlapping with g_in
                  if ( // WIC criteria… )
                        if (edit_distance(g_out , g_in ) < ε-dist)
                           // add to WIC
      if ( // WIC set criteria (i.e. support and maximality) )
            allWIC.add( WIC )
return allWIC          // all eps-similar variants
```

Figure 4.8: ε-similar aggregation algorithm

Following the pseudocode in Figure 4.8, grams are selected from the trie in an outer loop g_{Out}, while an inner loop iterates grams, g_{In} in the trie to find a *WIC*. ε-similar CWI have the property of $d(g_{In}, g_{Out})$ < ε-dist but, because ε-similar instances can have different lengths, grams in neighbor tries of two times ε-dist (above and below *n*) must also be searched and compared. Comparison utilizes edit distance because it allows for sequence misalignment and substitution, deletion, insertion and transposition operations cost 1. Another distance function is hamming distance, which only counts substitution operations. Hamming distance will only find optimally aligned workflow behaviors that only contain substituted actions or differ by inserted or deleted actions in latter parts of the sequence. Longest Common Substring (LCS) distance counts insertions and deletions but not substitutions. This means that substitutions will cost 2 instead of 1,

possibly removing the workflow instance from the instance set. Edit distance, though costly, was chosen to avoid introducing incompleteness before modeling.

The algorithm is enhanced in various ways to improve performance in the search. First, the search starts with the highest gram set. Second, repeat instance sets are avoided by only checking one subsequence among common subsequences. Also, if the trie is traversed in order, the discovery of substrings with edit distances above ε may allow one to branch back and skip leaf nodes in the trie. Certain scoring criteria may also be employed in this stage depending on their weighting, whether before the n-gram comparison or before the instance set is added to all instances. At first, it may appear that intermediate non-branching nodes in the suffix trie can also be collapsed to show shared substrings, which could but not necessarily describe a *WIC*. However, knowledge of the first activities can greater reduce search time by confining search to branches of the trie (or last activities by using a prefix trie).

4.5. WIC Rank

WIC Rank is a two stage algorithm involving a genetic algorithm for configuration of weights and a ranking function to implement criteria. Then WIC Rank performs *workflow behavior interest ranking* using a workflow oriented criteria and ranking. The purpose of this algorithm is to return relevant results based on user defined scoring criteria.

4.5.1. Workflow Behavior Interest Ranking

Using the ranking function in Section III, an ‘interestingness’ score is calculated. The ranking function is a weighted summation over the criteria of the *WIC*, so the

algorithm is not shown. However, a handful of scoring metrics are employed in our implementation of WIC-Rank. Here is a description of the criteria with rationale and formulas.

Cardinality. The number of instances in the instance set. For modeling, more instances will mean better models if they are complete representations. |WIC|

Common Start(Finish). The number of instances that have the same starting symbol as the seed divided by the total in the set. Computer behaviors often begin(end) with the same starting(finishing) sequence (i.e. logging in, checking e-mail).

Exactness. The ratio of the number of instances equal to the seed over the total number of instances is also important for the same reason as closeness. Exact matches would indicate less entropy in the set. $\frac{|ED\ (CWI_h,CWI)=0\ |}{|WIC|-1}$

Closeness. The average Hamming distance provides how close aligned instances are. It stands to reason that workflow instances of common sub-paths would be aligned. Since Hamming Distance was used in an earlier step to calculate tandem repeats, this information is already be available. $\frac{1}{|WIC|}\sum_{CWI\in WIC} HD(\ CWI_h, CWI)$

Tightness. Calculate the average edit distance in the set, which may indicate less noise between instances. Edit distance was calculated in WIC-Extract, and could be stored for this criteria. $\frac{1}{|WIC|}\sum_{CWI\in WIC} ED(\ CWI_h, CWI)$

Pairwise LCS. This captures all shared versus unique symbols and is calculated as the length of the pairwise *LCS* over the sum of all unique characters per instance.

Other semantics of the individual actions (i.e. specific action, action at an index) may be applicable. The result of workflow behavior interest extraction is a set of instances that could be selected as workflow instances in modeling applications, but the results need to be ranked to be presented to a process analyst.

4.5.2. Genetic Algorithm Configuration

Weight configuration is performed by a genetic algorithm (GA) that uses precision and recall metrics as a fitness function. An oracle WIC is presented as a target and weights are mutated by the GA to rank the oracle within the top 5 results. The weights are used by the ranking function.

In this discussion, the focus is on the alterations made to the GA to work for WIC weight configuration. The interested reader should refer to [76] for a background of GAs. In the first part, the GA initializes by assigning random weights to new candidates. The mutation function of the weights is subject to a uniform random process. The set of all WIC rankings are provided to a new candidate with the WIC criteria. With criteria, the initialization need not have the collection itself. The second part of the GA involves checking the fitness function of candidates. The fitness functions used in our GA are adapted from precision and recall metrics for information retrieval algorithms [77].

$$WIC\ Precision\ = \frac{|\ CWI\ \in WIC_o \cap CWI\ \in WIC_i|}{|\ CWI\ \in WIC_c\ |}$$

$$WIC\ \ Recall\ = \frac{|\ CWI\ \in WIC_o \cap CWI\ \in WIC_i|}{|\ CWI\ \in WIC_o\ |}$$

WIC precision identifies the weighted intersection of relevant CWI and retrieved CWI as the existence of CWI in the oracle WIC in the first five results of the candidate. Each index is scored and inversely weighted according to its ranking. Total and average fitness of the candidate population is calculated. This is monitored by a separate check to see if the GA is converging or diverging from a solution. The GA may stop if the stopping threshold is reached or maxRounds is reached. The pseudocode for this process can be seen in Figure 4.9.

```
// may stop early if the algorithm does not converge
while k < maxRounds:
    for candidate in cList {
        //compute fitness with precision and recall
        candidate.fitness = computeFitness(candidate.Ranked_WIC)
        //sum total fitness for average calculation
        totalFitness = totalFitness + candidate.fitness
    }
    // a tracking tool monitors average fitness for convergence
    averageFitness = totalFitness/len(candidateList)
    // determine if the answer has been reached
    bestCandidate = max(fitness, candidateList)
    if bestCandidate.fitness ≥ stoppingCondition
        return bestCandidate.weights
```

Figure 4.9: Fitness function update step and stopping condition check

If the GA does not stop, then the next generation is formed by selection of elites and creation of new candidates by mutation from the previous generation (Figure 4.10). Mutation occurs with higher probability of selecting genes from the more successful candidate ranking functions, but the randomness allows for greater exploration of the search space.

```
//...continued processing for next generation candidates
else{
        // select the next round of candidates from select elites
        // select elites do not change weights
        nextGenList = findElites(candidateList)
        // create the next generation of candidates by
                    // mutating weights of the candidates
        nextGenList.add(generateChildren(cList))
        cList = nextGenList
}
```

Figure 4.10: Selection of next generation candidates

The GA returns a set of weights to the process analyst to configure the ranking function to select interesting candidates that are somewhat to the process analysts query. The ranking algorithm produces a weighted score for each WIC using the WIC criteria and weights. The results and evaluation of the WIC algorithms are discussed next.

4.6. Evaluation of WIC-Extract and WIC-Rank

The WIC-Extract and WIC-Rank algorithms are varied in their approaches, applying from string processing to information retrieval and evolutionary algorithms. The algorithms provide a set of WIC that a process analyst can select from, easing his task of extraction from raw event data. To evaluate WIC Extract and Rank for workflow mining tasks this study considers complexity, performance and effectiveness aspects using complexity analysis, performance measurement and repeated trials of extraction and discovery of WIC in noisy continuous activity sequences. In addition, a simulated battle rhythm driven workflow is developed and monitored to show its viability for workflow behavior auditing in mission centric systems.

4.6.1. Complexity

The complexity of WIC Extract is dependent on the length of the input sequence |S|. The theoretic worst-case time complexity of repetition stemming is O(nk log(n/k)) where k is the maximal distance between two repeat copies (*miss*). Hamming distance functions have complexity O(log(n/k)) [78].

A vital step in the algorithm records all possible substrings (q-grams) from the sequence, but since this step is combined with the repetition stemming step in the algorithms the performance impact could be lessened. However, the worst case run time for generating all substrings in the input sequence is realized there are no tandem repeats in the input sequence. In this scenario, the first pass will produce 2 substrings each equal to a half of the input sequence and subsequent passes will produce 1 additional substring until the n^{th} pass, which produces n substrings. The dominating terms of this process results in $O(n^2)$ run time complexity.

The resulting suffix trie will contain CWI that have no exact matches and CWI with exact matches. The complexity of ε-similarity aggregation depends on the outer loop of the algorithm, iterating the number of CWI in the trie with exact matches, $m = |CWI_h|$. The middle loop of the algorithm checks similarity against all sequences within a bounded distance $\pm\varepsilon$ of string length, q. The inner loop iterates a maximum of $\frac{n}{2q}$ sequences for each string length and uses the edit distance for matching.

The implementation of an efficient edit distance function can be supported by the fact that matching is bounded by the similarity property. Under this condition, Ukonnen's algorithm [79] can compute edit distance in $O(q\varepsilon)$ time and $O(\varepsilon)$ space.

Myers approach using bit vectors [80] improves this result to $O(\sigma + \lceil q_1/w \rceil\ q_2)$ where q_1 and q_2 are two strings such that $q_1 \leq q_2$ and w is a bit vector size. Mapping a string into a bit vector parallelizes the dynamic programming matrix of the classical edit solution $O(n^2)$ and assumes σ as a bound on the alphabet size. Hyyro advances this to $O(\sigma + \lceil \varepsilon/w \rceil\ q_1)$ by bounding errors [80]. Considering these factors ε-similarity aggregation time complexity is no greater than $O\left(m * 2\varepsilon * \frac{n}{2q} * q\varepsilon\right) = O(nm\varepsilon^2)$ using Ukonnen's algorithm. Thus the worst case runtime complexity of the entire WIC-Extract algorithm is $O(n^2 + nm\varepsilon^2)\ or\ O(n^2)$. Size complexity is based on the size of the suffix trie, which is also $O(n^2)$. The only way to improve the result is to select heuristics to reduce the number of q-grams to create, which can be done with knowledge of the first or last activities in a target workflow instance. WIC Rank is not considered in complexity analysis since the workflow criteria can be harvested from distance based scores that happen within WIC Extract.

4.6.2. Performance

A Python implementation of WIC-Extract and WIC-Rank shows the capability of the algorithms to extract workflow instances from noisy and continuous activity sequences. Running on Linux with 1024 MB memory on a single 2.00 Ghz CPU, the algorithm was tested to handle 6000 events in 10 minutes on this modest platform searching for workflow instances between 7 and 11 sequences long. If users perform actions every 10 seconds, a population of 100 users could be continually monitored. With additional filtering and heuristics, porting to more powerful platforms and programming languages, these performance characteristics will improve.

To evaluate the returned results, classic precision and recall metrics are used [77]. Two reference strings were created from the workflows in the simulation and translated into the symbols used by the algorithm. Table 4.1 summarizes the returns of the top 5 results in the simulation. A genetic algorithm was used to choose weights for the ranking criteria using the precision and recall on the reference strings as a fitness function. In our experimental runs, exactness (most frequent precise patterns) and common start and finish behaviors were favored by the genetic algorithm, though this should not be considered a general result.

Table 4.1: Precision and recall results from WIC-Rank

	Reference Workflow 1		**Reference Workflow 2**	
Cluster Center	**Precision**	**Recall**	**Precision**	**Recall**
1) 'pqostuvw'	0.000	0.000	0.375	0.250
2) 'bcfehjil'	1.000	1.000	0.000	0.000
3) 'cpqoszuwp'	0.000	0.000	0.750	0.417
4) 'cpqoszuvw'	0.000	0.000	1.000	0.833
5) 'cpqoszuvwpx'	0.000	0.000	0.692	0.250

4.6.3. Effectiveness

As a source of usage data, we ran 39 workflow simulations where users primarily executed 2 scripted workflows from the Convoy Planning System. In the hypothetical mission-centric system, operators are responsible for selecting convoy routes and allocating crew members, cargo, and trucks along those routes. The users were also encouraged to perform the scripted actions out of order when the task performance

allowed it to be possible. The users also performed actions that were not task related at all to introduce noise into the event log. The result was an event log of 600 actions. To make the results understandable, the reader should be aware of an oracle CWI, '*bcfehjil*' which corresponds to a symbolically represented and stemmed subsequence in the log. The meaning of these symbols is not vital to understand the algorithm's effectiveness, though the reader should be assured that the map to activity labels in an event log.

1) CWI_h: cfehjil, LCS: ch, Score: 172.706410256

{'Distance': 3.5375, 'Common_Start': 1.0, 'LCS_Length': 2, 'Edit': 1.2307692307692308, 'Grapheme': 2.3, 'Common_Finish': 0.3076923076923077, 'Avg_Variant_Length': 7.075, 'Set_Size': 40, 'Hamming': 2.1794871794871793, 'Exactness': 0.9230769230769231}

[*cfehjil, cehji, cfehij, cfehil, cfehij,.....*]

2) CWI_h: bcfehji, LCS: bch, Score: 170.416666667

{'Distance': 2.3063063063063063, 'Common_Start': 1.0, 'LCS_Length': 3, 'Edit': 0.9722222222222222, 'Grapheme': 3.135135135135135, 'Common_Finish': 0.3888888888888889, 'Avg_Variant_Length': 6.918918918918919, 'Set_Size': 37, 'Hamming': 1.5833333333333333, 'Exactness': 1.0}

[*bcfehji, bcehj, bcfehi, bcfehi, bcfehi, ...*]

3) CWI_h: bcfehjil, LCS: bch, Score: 167.248076923

{'Distance': 2.6916666666666664, 'Common_Start': 1.0, 'LCS_Length': 3, 'Edit': 1.2307692307692308, 'Grapheme': 2.3, 'Common_Finish': 0.3076923076923077, 'Avg_Variant_Length': 8.075, 'Set_Size': 40, 'Hamming': 2.1794871794871793, 'Exactness': 0.9230769230769231}

[*bcfehjil, bcehji, bcfehij, bcfehij, bcfehij, ...*]

Figure 4.11: Ranked WIC with collection heads CWI_h

Using the segmented log L_{E1} from the WSD algorithm, WIC Extract and Rank produced ranked WIC. The top three results of the test is shown in Figure 4.11, though a process analyst would be given the results in a suitable format or visualization to make a decision based on these sets and information. The point of Figure 4.11 is to show how

the oracle has been identified within the top results of the search. It is logical that these three are selected, since the scores in the top 15 (Figure 4.12) exhibit a definite elbow beyond the first three.

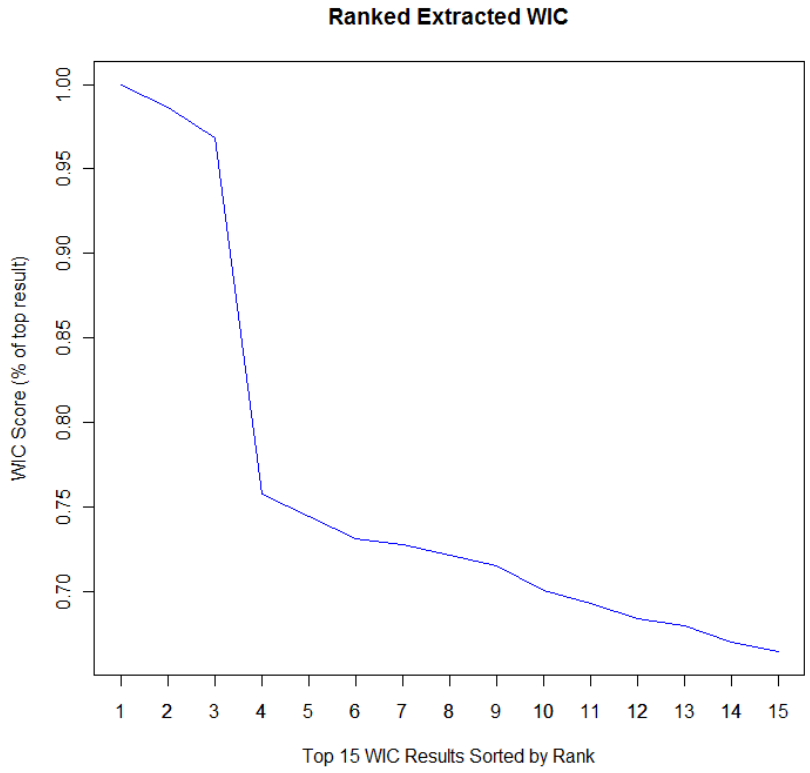

Figure 4.12: Top 15 scores for Ranked WIC.

Using the α-algorithm without any noise reduction, the discovered Petri net from the WIC ranked as the third result is shown in Figure 4.13. The graph should be compared to the ones produced out the outset of the section, where noise was interleaved with process instances. Noise has been left in the WIC clusters in the figure to show how the noise is no longer modeled as being interleaved, but instead pushed to the edges (in

this case, the finishing edge) of the workflow. This makes the result visually acute for a process analyst who can interpret the graph and the presence of the noise.

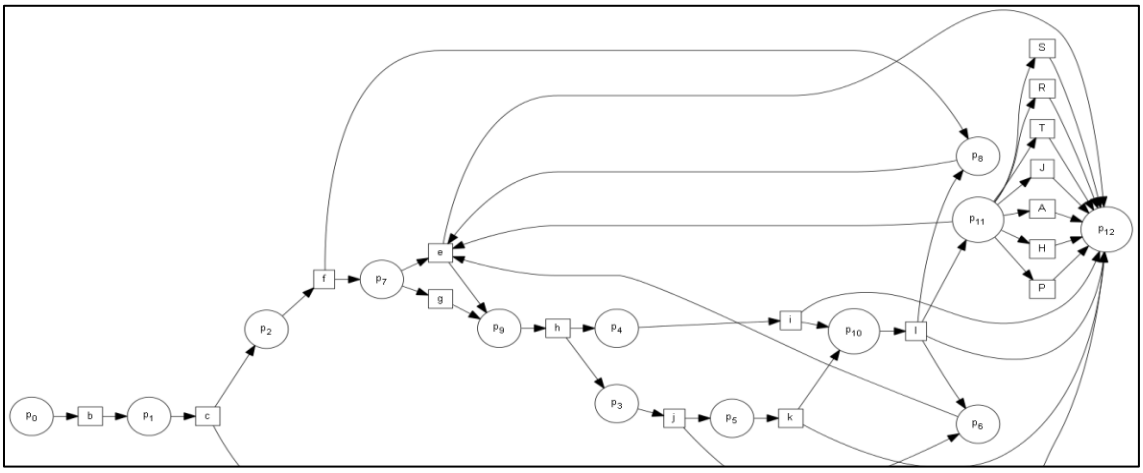

Figure 4.13: Petri net (w/ noise) discovered from third WIC result

4.7 Conclusion

The value of WIC Extract and Rank will ultimately be determined by their ability to create models that are appropriate to the task of the process analyst. In developing initial models, we are also able to indicate the viability of auditing structures of the RAMP and Releasability Gateways. Figure 4.14 showcases multiple artifacts created from the behavior instances. To start, the text cloud can give policy makers an intuitive sense of trending resource usage (1). Regular expressions (2) may be deployed at a network boundary or IDS. Other models may analyze action frequency with a moving baseline (3) of various aspects of the actions within the set of behavior instances. State

transition and frequency analysis may also be used in colored workflow transition systems (4) that can carry richer context than regular expressions for risk awareness.

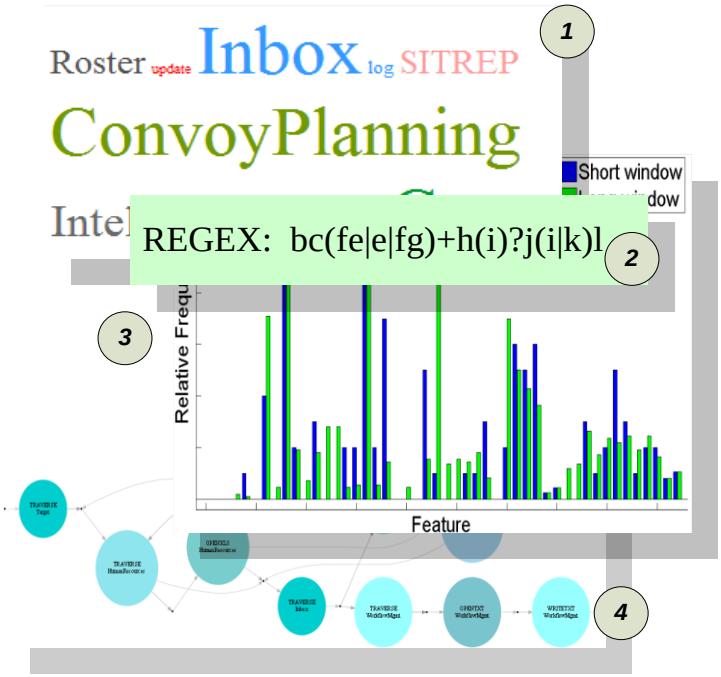

Figure 4.14: Process mining analytics

In our work in [81], we concluded that the facts that these models could be discovered suggest that situational awareness of workflow behaviors could conceivably be used for adaptation and risk management. In order to justify this claim, robust process discovery and workflow models are required. In mission-centric systems, the extraction of activity event logs from packet traces is problematic because of noise and loss during activity inference. However, distributed systems must also contend with temporal uncertainties in terms of timestamp inaccuracy.

5. SEMI-AUTOMATED AND DISTRIBUTED WORKFLOW MODELING

Business activities are increasingly automated as service oriented architectures support inter-organizational service querying, processing and transactions. As underlying technologies of virtualization and distributed computing have come to maturity, business and government is adapting to move into cloud business environment, collaborative environments that can be characterized by massive scale, service elasticity and resource sharing [22]. Human actors and software agents from several organizations may contribute to parts of a workflow or business process creating new opportunities for modern auditing. To maximize the likelihood that workflow behavior auditing will yield mission situational awareness, these opportunities must have technical solutions.

One such opportunity is process discovery and conformance checking of event logs from semi automated and inter-organizational workflows. Semi-automated workflows mean that activities performed by system to system interaction will be executed within seconds, while human activities could be accomplished with minutes, hours and longer time intervals. Inter-organizational workflows cause workflow execution and event logging to fall under the axioms of distributed system, particularly with respect to clock inaccuracy. The immediate consequence is timestamp inaccuracy, which coupled with rapid automated transactions can lead to uncertain ordering of events. Since process mining control flow algorithms depend on accurate causal ordering to produce process models, the effectiveness of process discovery will degrade as timestamp inaccuracy increases.

5.1. Timestamp Distortion

Therefore, to extend process mining, we explore the effects of timestamp inaccuracy by introducing the concept of timestamp distortion. In this section, a timestamp distortion technique is used for simulation of activity reordering to develop an appropriate algorithm. Timestamp distortion analysis gives the reader an understanding of effects that timestamp distortion has on process models leading to the desired solution for semi-automated, inter-organizational workflow management. A specific problem to address under these conditions is degrading behavioral conformance that reduces the usefulness of the models for conformance checking. As shown, perceived concurrency leads to state explosion and broken conformance checking in Petri nets. To address high concurrency, one must develop a model that capitalizes on simplicity of Flower Nets [5], but combats their over-generalization of behavior.

To compensate for the allowed by the flower model, the goal is to create a chain of flowers that will have better conformance checking properties than the flower Petri net, and while still retaining its minimalist nature. The formulation allows for admission control of the flowers into the model, so the amount of abstraction can be tailored to meet a goal FN/FP rate or minimal structure. In addition, new conformance checking tools are incorporated into the data structure, to establish checkpoints for the global system and curfews that limit activity execution within the flower structure. The model is evaluated against Heuristic miners, Flower models and Hidden Markov Models. First, the need is presented for such models in workflows that suffer from timestamp distortion.

5.2. Process Mining with Timestamp Distortion

The approach to further auditing technology for cloud environments is through process mining of event logs, but these methods cannot be applied without understanding logs in asynchronous distributed systems. We developed the Workflow Auditing and Simulation Platform (WASP) to simulate reordering in event logs in semi-automated workflows where activity wait and service times occur on the order of milliseconds to minutes. The WASP present a novel approach to simulating timestamp distortion for process mining event logs that undermines limiting assumptions. The analysis describes the effects on causal reordering effect on logs and its subsequent effects of reduced conformance and state explosion in process models.

Semi-automated and inter-organizational workflows may lead to the perception of timestamp distortion. Process execution imposes a natural ordering of events, but distributed activity execution and distributed log writing allow the possibility that events are recorded out of order [61]. A global network may have a high amount of timestamp error due to inaccurate clocks on hosts [39]. Correlation of logs from heterogeneous sources may also inject error in timestamps [82]. Another source of time offsets is clock drift where clocks, possibly synchronized at one time, drift away from a master time due to imperfection of hardware oscillators or ambient physical effects [83][84]. As imprecise and inaccurate timestamps are among the top five data quality issues in process mining [63][64], timestamp distortion is an important issue to address.

5.2.1. Simulation of Timestamp Distortion

We describe a Gaussian-Inaccurate clock model supports the simulation approach for understanding the effects of timestamp distortion in process mining. A running example of a Credit Application Workflow provides realistic guide for analysis. The study points to behavioral conformance issues stemming from a mishandling of concurrency in discovered Petri nets.

5.2.1.1. Gaussian Inaccurate Clocks

A log writing system can be modeled as an asynchronous message writing system where a stochastic process alters the true timestamp t_s of an activity execution to an inaccurate version t'_s. The causes of timestamp inaccuracy are a set of random variables U = { $u_1 \ldots u_n$ }. A timestamp may be distorted by an additive time adjustment to the true timestamp from these random phenomena. The calculation of a distorted timestamp t'_s from a true timestamp t_s for any given state is

$$t'_S = t_S + t_{u_1} + \cdots + t_{u_n} \; ; \; \textit{for all distortion rates } u_D \in \mathrm{U} \mid u_D = \{ u_1 \ldots u_n \}$$

where u_i is a stochastic process that creates an linear adjustment t_{u_n} for a timestamp. The effect of performing this adjustment over a sequence of events may induce a perceptive reordering that may affect the inference of causal dependency in process discovery and analysis techniques in process mining. To investigate this phenomenon, a sequence reordering algorithm simulates timestamp distortion in activity events and resorts the activities using the distorted timestamp. The *Reorder_Gaussian* algorithm alters timestamps by adding a time delta that is generated using a random Gaussian process.

The Gaussian process simulates the time delta from time kept by an imaginary master clock that holds the timestamp average. Consider that there are *N* computers in a network and each has a different clock. One may take the average of all clocks to get the mean time. Then, at any time step, the difference between the time from any clock and the average clock is calculated to obtain a single clock's time delta. The difference between the zero mean clock and itself is 0. To calculate the standard deviation of all clock deltas, the square root of the sum of all squared deviations of the clock deltas from the mean divided by the number of clocks in the system.

$$\sigma_c = \sqrt{\frac{1}{|C|}\sum_{n=1}^{|C|}[(c_n - \mu_c)^2]}$$

With a mean of 0 and a standard deviation that is variable during the simulation, one can increase the variance and scale of the random clock deviations using a Gaussian distribution for a random number generator.

Definition 5.1. (Gaussian Timestamp Distortion) *Let (μ_c) be a zero mean clock in an asynchronous system and let (σ_c) be the standard deviation of timestamps from the clock. Then timestamp delta (t_D) is a R.V. is a Gaussian timestamp distortion that fits the distribution*

$$pdf(t_D) = \frac{1}{\sqrt{2\pi{\sigma_c}^2}} e^{\frac{-(t_D-\mu_c)^2}{2{\sigma_c}^2}}$$

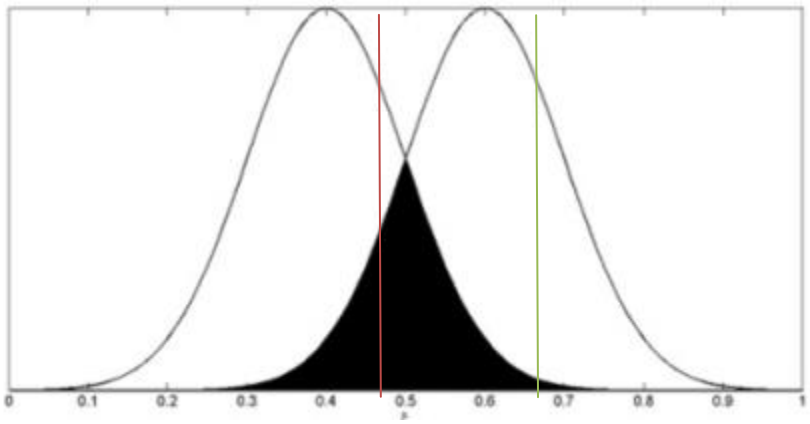

Figure 5.1: Potential for Gaussian timestamp distortion

Figure 5.1 illustrates the potential of reordering for two events under Gaussian timestamp distortion. Two activities are executed at the time of a true timestamp (a first activity (red) at 0.4, a second activity (green) at 0.6 seconds). When a timestamp adjustment is simulated using the Gaussian pdf, the overlap region shows the potential for the event to be reordered in a log. A low standard deviation will have a lower probability to reorder events in the system than a high standard deviation will result in high entropy in the system, because the tails of the Gaussian will have more overlap.

5.2.1.2. The Credit Application Workflow

A workflow model used in simulation in the process mining literature is the Credit Application Workflow (CAWF) [85]. The CAWF describes potential activity paths that can be taken to accomplish a task of approving a loan or line of credit. It starts with the '*receive application*' action, which triggers the execution of the workflow. From there, a token based model can route tasks to various people to execute tasks such

as *'check for completeness'*, *'make decision'* and *'notify'* acceptance or rejection. A visualization of CAWF is shown in Figure 5.2.

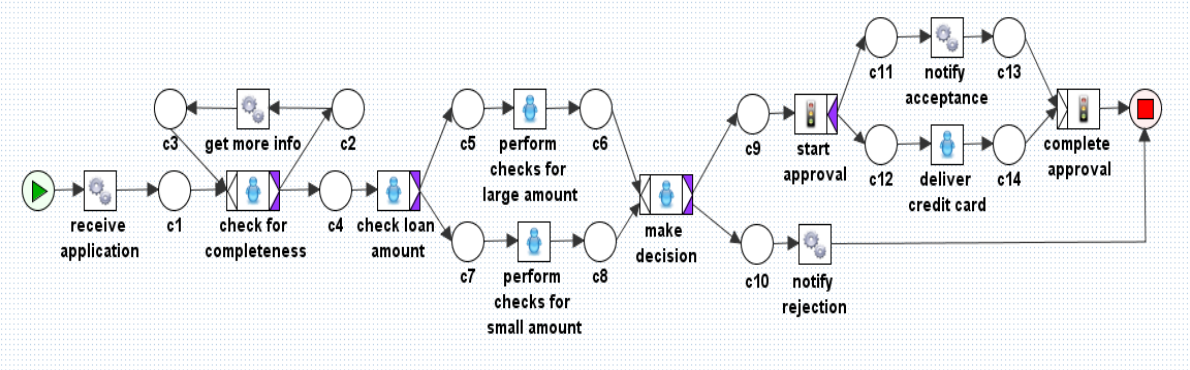

Figure 5.2: Credit application workflow

We used the WASP to generate logs of workflow instances of the CAWF. The CAWF event log describes potential activity paths that are taken through the model. Organizational information such as roles and personnel has been incorporated into the simulation as well historical information concerning the delays between activities and service times for activities. This results in a realistic simulation log, seen in Figure 5.3.

```
1205264860000,1421617319,B,TASK_receive_application_3,complete,loanAmt,3877,LewisF,2008-03-11T20:47:40.000+01:00,
1205272333000,1968082134,C,TASK_check_for_completeness_4,start,,,LewisC,2008-03-11T22:52:13.000+01:00, 1205274336000,-
1379851195,D,TASK_check_for_completeness_4,complete,completeApp,false,LewisC,2008-03-11T23:25:36.000+01:00,
1205275020000,-1945361321,E,TASK_get_more_info_5,start,,,LewisC,2008-03-11T23:37:00.000+01:00,
1205278521000,113430820,F,TASK_get_more_info_5,complete,,,LewisC,2008-03-12T00:35:21.000+01:00,
1205296343000,1968082134,C,TASK_check_for_completeness_4,start,,,GoldB,2008-03-12T05:32:23.000+01:00, 1205298502000,-
1379851195,D,TASK_check_for_completeness_4,complete,completeApp,true,GoldB,2008-03-12T06:08:22.000+01:00,
1205299821000,39054517,G,TASK_check_loan_amount_6,start,,,LewisF,2008-03-12T06:30:21.000+01:00,
1205300313000,1915798534,H,TASK_check_loan_amount_6,complete,,,LewisF,2008-03-12T06:38:33.000+01:00,
1205301952000,1603654300,I,TASK_perform_checks_for_large_amount_7,start,,,JonesF,2008-03-12T07:05:52.000+01:00,
```

Figure 5.3: Event log gathered from simulation

We made the assumption that automation in a service oriented environment would make activities in the event logs significantly faster, so service times and wait times are $1/1000^{th}$ of the speed of their human counterparts. Process mining has not considered the difficulties of reordered events because these effects are barely noticeable with human agents. With fully automated software agents in distributed environments, these assumptions will no longer hold.

5.2.1.3. Process Discovery with Timestamp Distortion

Timestamp distortion is detrimental because it can lead to different perspectives on the ordering of activities by different observers. The core focus of control flow algorithms is determining *causality*. C*ausality* is the relation between two events in sequence where one is understood or perceived to be the consequence of the other. Causality is basically depicted as atomic rules, where the observation of a_1 followed by a_2 is evidence to support the claim that a_1 caused a_2 or $a_1 \rightarrow a_2$. Yet it is a claim and not fact. The claim $a_1 \rightarrow a_2$ can never be proven [61] because of the potential for a counterexample, yet process mining assumes that the relationship is supported by observation in event logs.

By discovering transitions, places and flows, a Petri net can be formed that closely resembles the original workflow (Figure 5.4 and 5.5). By visual inspection, one can see the similarities between the two models. We superimpose symbols on the original log to collapse their representation so the resulting process model may be understood.

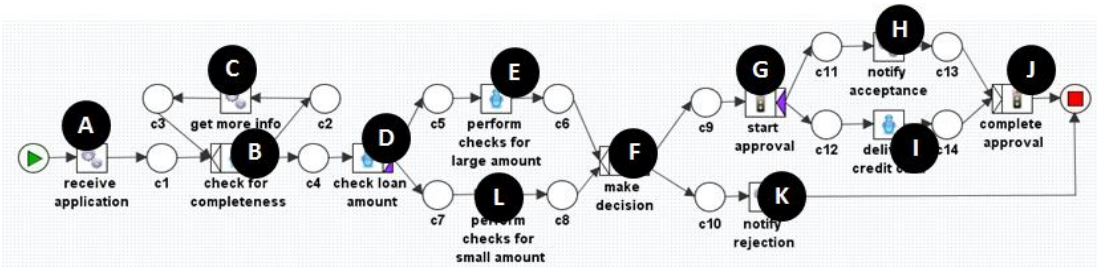

Figure 5.4: Symbolic mapping of CAWF activities to transitions

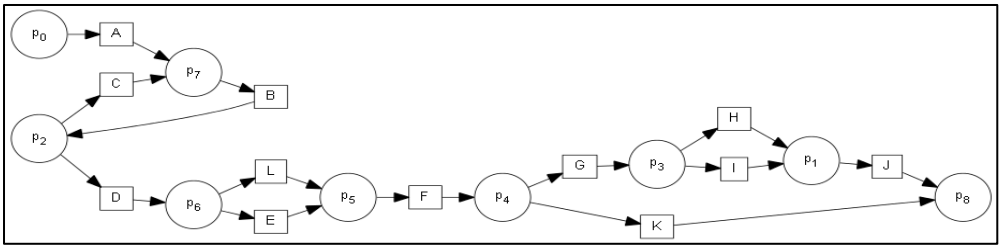

Figure 5.5: Discovered workflow without timestamp distortion

Visual inspection is adequate if it can be seen that the models conform to the original exactly, but process models formed with timestamp distortion (Figure 5.6) require a quantified measurement. In the figure, timestamp distortion affects the process discovery algorithm because causal dependency relations have been changed.

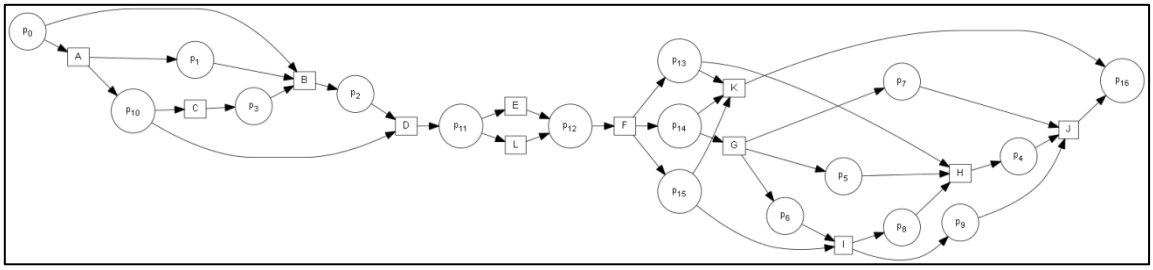

Figure 5.6: Discovered workflow with timestamp distortion

5.2.2. Behavioral Conformance with Timestamp Distortion

Conformance checking can help us evaluate the discovered process model for an intended purpose. The purpose is to support detection of appropriate and inappropriate behavior in semi-automated and inter-organizational workflows. This requires models that will detect despite the effects of timestamp distortion. Behavioral conformance analysis in discovered Petri nets to expose the problem of timestamp distortion and points the reader to the desired solution.

5.2.2.1. Token Replay

Behavioral conformance measures the difference between the expected behaviors of two Workflow Nets. There are a number of ways to do this, but here the focus is on methods that involve token replay [5]. Token replay describes the parsing action of a Petri net (PN) over a given input sequence. As an illustration, consider the parsing sequence in Figure 5.7. (1) An event log may replay on this Petri net with an initial marking at p_0. The transition after the first place is *enabled*. (2) An initial activity is parsed from any sequence in the event log, which in this case is *a.* The activity maps to transition '*a*' in the PN and the transition fires since it is enabled. The firing rule does two things, first it *consumes* the token at the input place and second *produces* a token for every output place. (3) Three transitions are enabled after '*a*' fires. Depending on the sequence, either '*e*' can fire by consuming the tokens at *p1* and *p2* or '*b*' and '*c*' can independently fire each consuming the tokens at p_1 and p_2, respectively. In either case, two tokens are produced at p_3 and p_4 (4). With '*d*' enabled, it can also fire when it is parsed and mapped, completing the firing sequence.

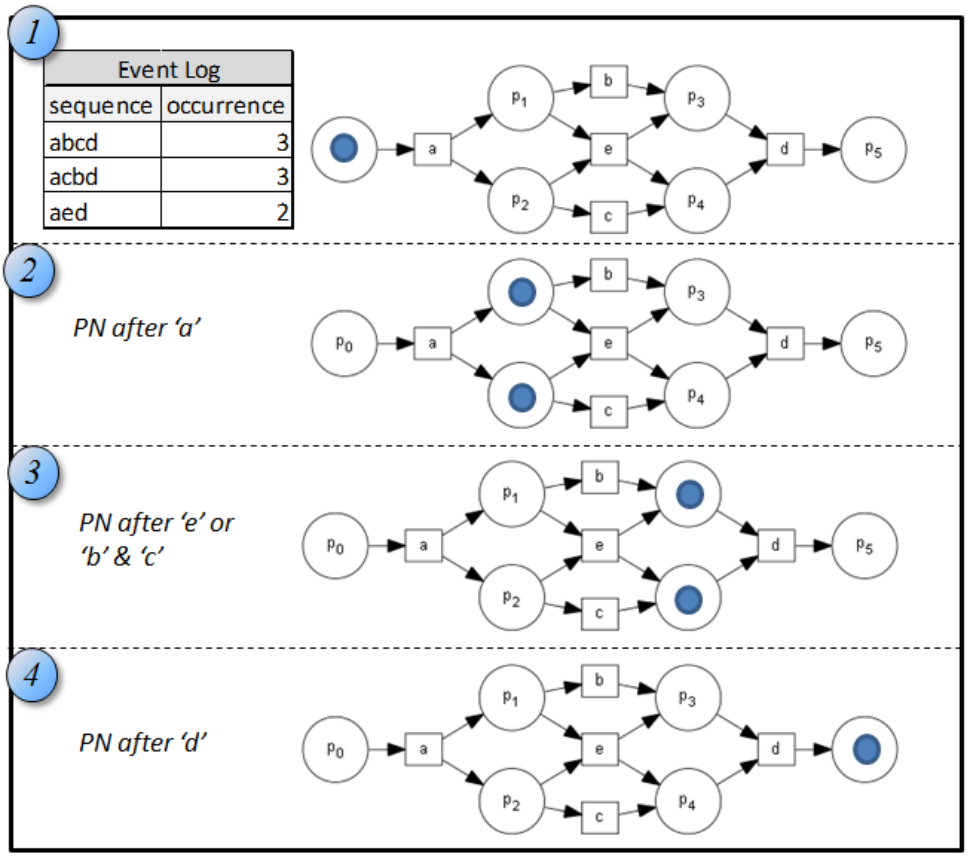

Figure 5.7: Token replay from a simple event log

The instances provided in the illustration a parsed exactly to the specification of the Petri net. It is possible that this is not the case. For example, any sequence of the above that did not start with '*a*' would have been identified as a *miss*, since the activity was parsed from the sequence but could not fire. Furthermore, any sequence not ending with '*d*' would have been considered incomplete due to the fact that a token is *remaining* after the complete parsing of the activity sequence.

Token replay can be used to estimate behavioral conformance of a given collection of activity sequences to a model. Using a Boolean replay function $PN(\sigma_i)$, one

would count sequences, $\sigma_i \in \sigma$, that had complete parsings and left no remaining tokens, $PN(\sigma_i) = 1$. Similarly, sequences that could not be replayed in this way $PN(\sigma_i) = 0$. If the sequences σ are considered to be acceptable instances, then the PN should parse the entire sequence without missing or remaining tokens. If the PN does not accept them, then this is a false positive, $PN(\sigma_i) = 0$. Therefore, a measure of the false positive rate for any discovered PN is defined:

Definition 5.2. (False Positive Rate) *Let* $\sigma = \{ \sigma_1 \ldots \sigma_i \ldots \sigma_n \}$ *be a set of activity sequences, PN a Petri net and* $PN(\sigma_i)$ *be a Boolean replay function of PN, then the False Positive Rate (FP) for a given Petri net is*

$$FP(\sigma, PN) = \frac{\sum_{i=1}^{n} [PN(\sigma_i) = 0\,]}{|\sigma|}$$

To define a false negative rate, a set of erroneous or faulty activity sequences γ must be available. The negative set of sequences should be rejected by a model, $PN(\gamma_i) = 0$. If it is not, then a false negative occurs because it replays the entire negative sequence, $PN(\gamma_i) = 1$.

Definition 5.3. (False Negative Rate) *Let* $\gamma = \{ \gamma_1 \ldots \gamma_i \ldots \gamma_n \}$ *be a set of activity sequences, PN a Petri net and* $PN(\sigma_i)$ *be a Boolean replay function of PN, then the False Negative Rate (FN) for a given Petri net is*

$$FN(\gamma, PN) = \frac{\sum_{i=1}^{n} [PN(\gamma_i) = 1\,]}{|\gamma|}$$

Another way to measure behavioral conformance is through the approximate matching of sequences using ratios of produced (p) to consumed (c) and missing (m) to remaining (r) tokens. This is called fitness since it refers to the ability of the *PN* to fit the observed behavior in the log.

$$fitness(\sigma, PN) = \frac{1}{2}\left(1 - \frac{m}{c}\right) + \frac{1}{2}\left(1 - \frac{r}{p}\right)$$

The fitness metric only uses positive sequences, since they are used to measure the model ability to capture all real behavior.

5.2.2.2. Token Replay Analysis of Timestamp Distortion

A discovered Petri net like the one in Figure 5.6 was formed in a timestamp distorted environment. Given the altered structure of the Petri net, it is a reasonable claim that the behavior of this log no longer conforms to the oracle. This is problematic if an adaptive system deploys a discovered workflow behavior model to detect positive or negative activity sequences. The analysis is evidence to support this claim and identify a cause of poor detection.

Figure 5.8 shows FP rates for the discovered Petri net using a heuristic miner. With clock variance ranging from 0 to 2.5 ms, the model discovered with the Heuristic miner degraded abruptly at increasing levels of timestamp distortion. Timestamp inaccuracy is measured in terms of variance from a zero mean. When clock variance rose to 0.11 ms, a concurrent relationship was identified where previously there was not. As a result, the discovered model changed and the number of places grew to 15 and flows 38. Though the FP rate is already unacceptable for auditing purposes, the rate began to settle due to increased perturbation in the behavior becoming increasingly similar to the new

model in terms of conformance. However, variance makes the workflow model unstable at 1 ms and the model breaks. This is expected since the ability for a heuristic miner to work is limited by the existence of adequate statistical support for causal ordering. When concurrency is added to a model, conformance and detection degrade.

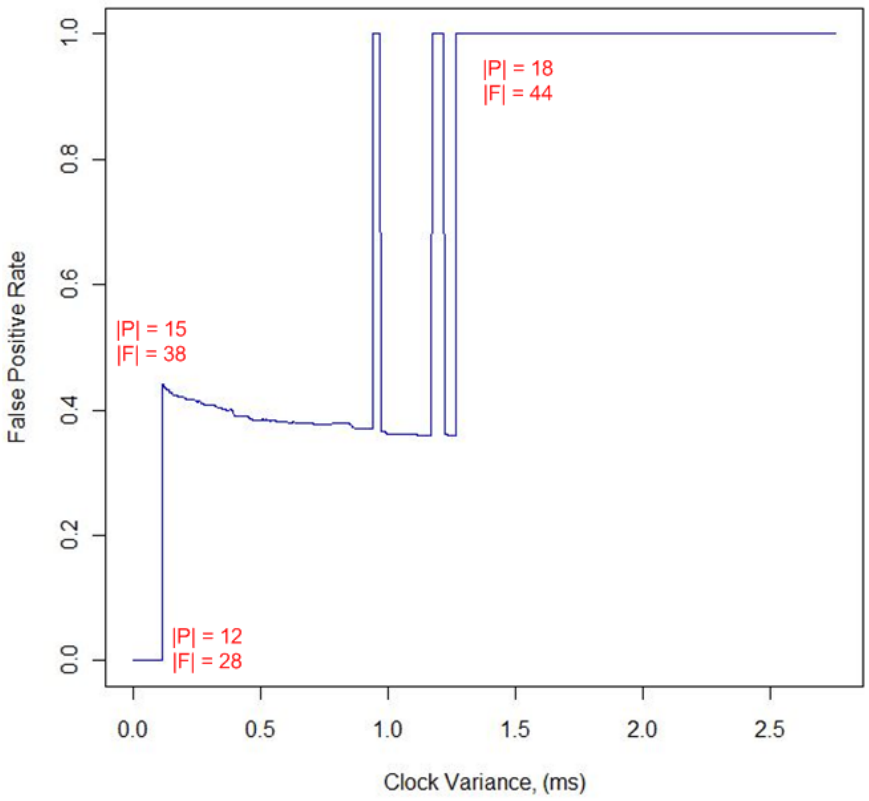

Figure 5.8: False positive rate for Heuristic mined Petri net

5.2.2.3. Concurrency-Conformance Tradeoff

An alternative Petri net model is the flower net in Figure 5.9. The Flower Petri Net (FPN) holds an advantage over the heuristic method for its stable construction in timestamp distorted environments. This leads to a consistent ability for the activities

{*a,b,c,d,e,f*} to replay in the FPN in any order so it is tolerant of out of sequence observations in an event log, so long as the start and end activities remain invariant and all activities parsed by the FPN are within the language of the FPN. Another benefit from the FPN is that the number of nodes in the petri net are limited to one input place, one output place and one flower place for any number of transitions. This limits the size of any flower model Petri net to $|N| = |T_i| + |T_o| + |T \setminus (T_i \cup T_o)| + 3$.

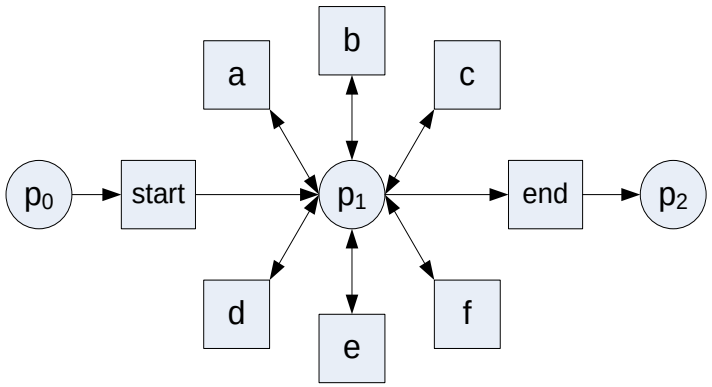

Figure 5.9: Flower Petri net.

FPN can greatly reduce the effects of concurrency, but they also have a disadvantage in that the exhibited behavior in the log may not encompass all possible activity sequences as this model does. This means that the behavioral conformance checking ability will be limited to detecting out of language activities. The false negative rate within the language of activity labels will be very high.

In a mission centric setting, semi-automated and distributed workflows will demand highly tolerant modeling techniques. Using FPN structures may help reduce the

effects of concurrency and improve behavioral conformance. In doing so, the benefit of low FP rates of FPN can be gained. At the same time, the goal is to decrease the FN rate as seen in Flower Net, by allowing the model to maintain invariant characteristics. The solution to this problem is the Flower Chain and Flower Chain Algorithm, which provides an incremental approach to grow or collapse a Petri net according to the environment.

5.3. Flower Chain Models

The need to establish modern workflow behavior auditing tools leads us to the development of the Flower Chain Net (FCN). The FCN is built upon the concepts in Petri nets to maintain their formal properties, but the formulation also adds checkpoints and curfews as tools to track the health of a workflow. Semi automated and distributed workflows will need additional tools to ensure compliance despite degradation from high concurrency. Here, we offer checkpoints and curfews, flower links and flower chain nets for this issue.

5.3.1. Checkpoints and Curfews

Checkpoints and curfews establish bounds on behavior exhibited in the log without expanding states. A checkpoint is applied in the FCN at bottlenecks in the model, which could be points of exchange between organizations or the completion of a sub task. Checkpoints are intended to handle the handoff of a human part of an automated workflow to a automated section. To do so, it must carry useful information is the time of arrival and behavior of workers before the checkpoint.

Definition 5.4. (Checkpoint) - *A checkpoint is a tuple CH = (**τ**,M,β_h, g_h) where τ are time bounds to reach or exit a checkpoint and m is the minimum and maximum firings of a Petri net before a checkpoint.* B_h, *is a tolerance ratio and* g_h *is a Boolean decision function such that:*

$$g_h(\tau, m) = [(1 - \beta_h) * M_{min} \leq m \leq M_{max} * (1 + \beta_h)]$$
$$\wedge [(1 - \beta_h) * \tau_{min} \leq \tau \leq \tau_{max} * (1 + \beta_h)]$$

A checkpoint will bound the behavior before entering a concurrent region while a curfew bounds the behavior within a concurrent region. A curfew is intended to monitor the automated parts of a workflow or activities that have uncertain observation. A curfew features rate and firing limits discerned from historical rates and limits in a training log.

Definition 5.5. (Curfew)- *A curfew is a tuple CF = (K, Λ,* β_a, g_a *) where K is a set of ordinal maximum and minimum firing limits within a concurrent region and Λ is a set of maximum and minimum firing rates.* β_a, *is a tolerance ratio and* g_a *is a Boolean decision function such that:*

$$g_a(k, \lambda) == [(1 - \beta_a) * \mathrm{K}_{min} \leq k \leq \mathrm{K}_{max} * (1 + \beta_a)]$$
$$\wedge [(1 - \beta_a) * \leq \Lambda_{min} \leq \lambda \leq \Lambda_{max} * (1 + \beta_a)]$$

Checkpoints and curfews form the basis of a detection capability, but the structural characteristics of FCN augment this by maintaining strong causal relationships.

5.3.2. Flower Chain Petri Nets

A flower chain model is composed of places with checkpoints and curfews as well as places without them. The places without them are under the definition for a workflow net, but the places with them fall under a definition of a complex place. That is the flower link, and it extends from the flower petri net model.

Definition 5.6. (Flower Link) - *A flower link, fl, is a 6 tuple, fl = (p_f, T_f, F, l, CH, CF)such that*

1. p_f *is a singleton flower place*
2. $t \in T_f \mid T_f = \{t_1, \ldots, t_m\}$ *is a finite and nonempty set of transitions*
3. $p_f \notin T$
4. $F \subseteq (p_f \times T) \cup (T \times p_f)$ *is a set of directed arcs*
5. $l : T \rightarrow \Gamma$ *is a labeling function of transitions T to a finite set of activity labels,* Γ
6. *CH (CF) is a checkpoint (curfew) function*

Flower links possess a desirable property of minimal states such the the number of nodes $|N| = |T| + 1$ and flows are also limited to $|T_f|$ within the structure. $|T_f|$ is also described as the petal size of the structure. Flower links can form chains, thus the name of the model. However, a simple composition does not allow for Petri net concurrency relationships, so a more general construction of the FCN requires it to be incorporated under a general definition for a Petri net.

Definition 5.7. (Flower Chain Net) - *A Flower Chain Net is a 6 tuple FCN = (P, T, F, FL, $M_{0,}$ l) where:*

1. *$FL = \{fl_1, \ldots, fl_k\}$ is a finite and nonempty set of flower links*
2. *$P = \{p_1, \ldots, p_n\} \cup \forall p_f \in FL$ is a finite and nonempty set of places*
3. *$T = \{t_1, \ldots, t_m\} \cup \forall T_f \in FL$ is a finite and nonempty set of transitions*
4. *$P \cap T = \emptyset$*
5. *$F \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs*
6. *$M_0 : P \rightarrow \mathbb{N}_0$ is the initial marking*
7. *$l : T \rightarrow \Gamma$ is a labeling function of transitions T to a finite set of activity labels, Γ*

The definition of the FCN allows for Petri net behaviors and properties of Workflow Nets. The FCN allows the potential to limit concurrent relationships to reduce complexity within the flower link. At the same time it can improve behavioral conformance over the flower petri net in terms of FN rate through checkpoints, curfews and a Petri net structure for strong causal dependency.

5.4. Flower Chain Discovery

FCN can be composed using the Flower Chain Discovery Algorithm (FCDA). The algorithm extends from the α-algorithm to incorporate flower links to reduce complexity in environments with timestamp distortion. The algorithm is a 4 phased construction of FCN. In the first stage, a footprint matrix, transition sets and identify flower candidates are formed. The second stage builds flower links and the third stage

grows the rest of the Petri net around them. The final stage sets checkpoints and curfews for replay.

5.4.1. Identifying Flower Candidates

FCDA starts by parsing an activity event log, L and assigning symbolic labels to them. The symbolic transformation distinguishes activities from transitions, as they are incorporated into the model. Four sets of transitions are created:

1) $T = \forall t \in L.$

2) $T_S = \forall \sigma \in L \ , \forall t \in T \ | \sigma \ = \{ \sigma_1 \ldots \sigma_n \} \ , t = \sigma_1$

3) $T_E = \forall \sigma \in L \ , \forall t \in T \ | \sigma \ = \{ \sigma_1 \ldots \sigma_n \} \ , t = \sigma_n$

4) $T_R = \forall \sigma \in L \ , \forall t \in T \ | \sigma \ = \{ \sigma_1 \ldots \sigma_n \} \ , t = \sigma_i = \sigma_{i+1}$

From T, the rule sets are inferred using log based ordering relations [2] for $a, b \in T$:

5) $R = \ \forall \ a > b$: a leads to b, $\exists \sigma \in L$, a $= \sigma_i$ and $b = \ \sigma_{i+1}$

6) $R_{\rightarrow} = \ \forall \ a \rightarrow b$: *(a > b) and ¬(b> a)*

7) $R_{||} = \ \forall \ a \ || \ b$: *(a > b) and (b> a)*

8) $R_{\#} = \ \forall \ a \ \# \ b$: *¬ (a > b) and ¬(b> a)*

The log ordering relations form the footprint matrix shown in Figure 5.10. FCN capitalizes on the minimalism in flower links to collapse concurrency. To identify flower link candidates, the footprint matrix is 1) scanned by row or column and 2) stored in a concurrency table in sorted order of the degree of cardinality of each concurrency set. This constitutes a set of transitions for a flower link candidate ($T_{fl} \in T_{FL}^{*}$):

9) $T_{fl} = \ \{a, b_1 \ldots b_{|T|} \}, 1 \leq i \leq |T| \ | \ a, b \in T \ \wedge \ a \ || \ b_i$

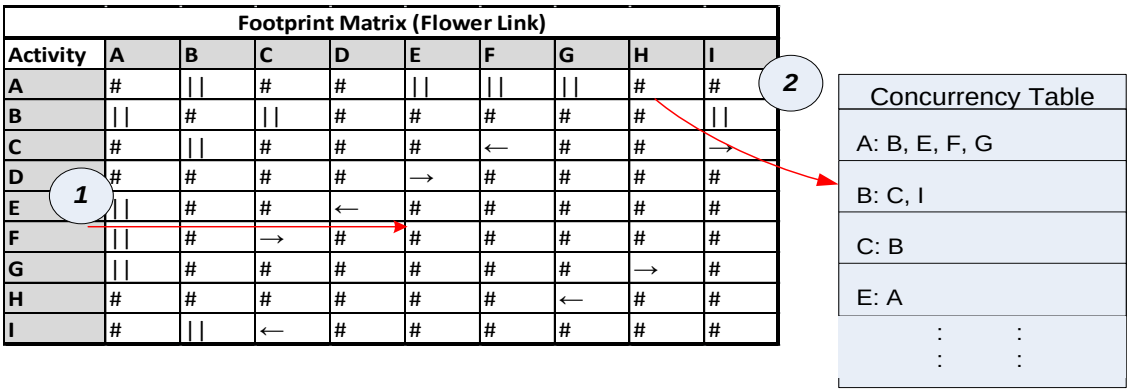

Figure 5.10: Finding flower link candidates from the footprint matrix

5.4.2. Building Flower Chain Networks

In the next stage, flower places are created and linked into the other places in FCN as the FCN gains its directed graph structure as candidate selection proceeds first. For FCDA and its purpose to improve conformance in discovered PN, it is logical to reduce the largest degree of concurrency first and collapse down as needed. FCDA has a tunable parameter *d* to limit the reduction of concurrency in the FCN.

10) $T_{FL} = T_{FL}^{*}, |T_{fl}| > d$

To build the flower links, p_f is created:

11) $FL = (p_f , T_{fl}) \mid \forall T_{fl} \in T_{FL} , p_f = \{ (T_{fl}, T_{fl}) \}$

Though it may not be evidenced by the log, all transitions in T_{fl} are put in input and output places of p_f. This is done to maintain generalization and keep FP low, but one could expand the set T_{fl} to identify input and output only sets if the situation requires it.

Step 11 builds all the flower links but the remainder of the place nodes must be identified before assigning flows.

12) $X' = (a, b) \mid \exists\ a \rightarrow b \in R_{\rightarrow}, a, b \notin T_{FL}$

13) $FL = (a, T_{fl}) \mid \exists\ a \rightarrow b \in R_{\rightarrow}, b \in T_{FL}$, $FL = (T_{fl}, b) \mid \exists\ a \rightarrow b \in R_{\rightarrow}$, a $\in T_{FL}$

The algorithm builds the rest of the places to find maximal sets, X'' that are recursively composed from X' under the condition that the sets of transitions A and B have no relationship to each other. The composition allows subsets to be marked and discarded from X''.

14) $X'' = (A , B) \mid A \subseteq T \wedge B \subseteq T \wedge \forall_{a \in A}\ \forall_{b \in B}$, $\exists X' = (A, B) \wedge A \# B$

15) $Y = (A , B) \mid \forall_{A'}\ \forall_{B'}\ (A , B) \not\subset (A' , B')$

The final place building step combines the flower links, maximal $R_{\rightarrow}$ places and the input and output places.

16) $P = p_{(A, B)} \mid (A, B) \in Y \cup FL \cup \{p(\emptyset, T_S), p(T_E, \emptyset)\}$

Flows are the set of all relationships between the places in P and transitions in T.

17) $F = (T , P) \cup (P , T) \mid (\forall_{p \in P}\ \exists_{b \in T}\ b \in p\bullet) \vee (\forall_{p \in P}\ \exists a_{\in T}\ a \in \bullet p)$

where the operator $\bullet p$ ($p\bullet$) denotes the input (output) transitions of p. Finally, the petri net is formed as

18) $FCN = (FL , P, T, F)$

5.4.3. Setting Checkpoints and Curfews

After the formation of the petri net, FCDA assigns values to *CH* and *CF* in *FL* in the following manner. A training log is replayed over FCN, counting tokens produced and consumed at each place and in the flower links. Timing information is recorded between *FL* and the maximum and minimum time periods are recorded, $\boldsymbol{\tau} = \{\tau_{min}, \tau_{max}\}$.

Similarly, the number of firings of the FCN between *FL* is record in $M = \{m_{min}, m_{max}\}$. The information is used to set the checkpoint, *CH*.

19) $\forall fl \in FL, fl_{CH} \leftarrow (\boldsymbol{\tau}, M, \beta_h)$

where β_a is a tunable tolerance parameter. The curfew is set in a similar fashion, as token firing and rate information is harvested during the replay Finally, CF is set using the maximum and minimum firings (rates) in the training period such that $K = \{k_{min}, k_{max}\}$ ($\Lambda = \{\lambda_{min}, \lambda_{max}\}$) and

20) $\forall fl \in FL, fl_{CF} \leftarrow (K, \Lambda, g_a)$.

The flower links provide detection tools in place of the expressiveness of concurrency modeling. The results of which will yield smaller and comprehensible graphs that can be used for auditing semi-automated and distributed workflows.

5.5. Results and Evaluation

Flower Chain Networks have structural and behavioral characteristics that are advantageous for semi automated and distributed workflows. The Flower Chain Discovery Algorithm is an extension within a family of α-algorithms [51] suited for a particular purpose of managing highly concurrent automated processes with the potential of timestamp distortion.

5.5.1. Structural Characteristics of Flower Chain Nets

The ability to scale and collapse concurrency results can set a bound on node size in the graphs. At the same time that FCN are able to reduce concurrency, they are also able to model global concurrent relationships. These properties can be seen through datasets with collapsible and global concurrency.

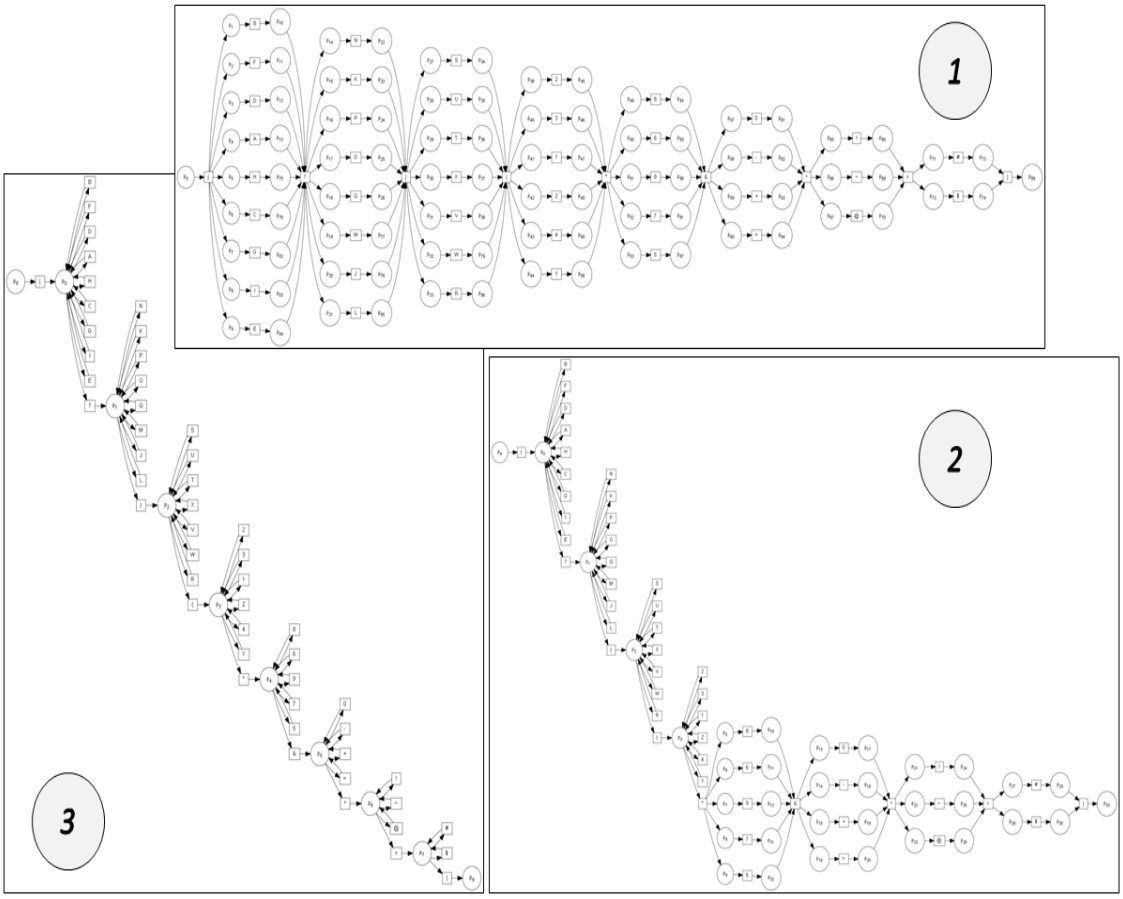

Figure 5.11: Collapsible concurrency with FCN

5.5.1.1. Collapsible Concurrency

Collapsible concurrency is a structural feature of FCN to make it possible to represent concurrency at various levels. Figure 5.11 shows a progression through three stages of a pedagogical event log with several sizes of concurrent groupings. The log was generated using sequences of random activities followed by an invariant activity, another random sequence with another invariant. Several sets were built so that concurrency may be perceived within subsections of an activity event log.

In the first stage, all concurrent relationships are visible. In the second stage, they reduce as flower link candidates require six members to maintain a concurrent region. At stage three, concurrency is completely collapsed. This is a useful property because it allows a process analyst to choose the proper level of abstraction for the modeling task they are assigned to. The concurrency reduction does not need to be top down. It may be bottom up or follow another candidate selection algorithm.

5.5.1.2. Bounded Size

The bounded size of FCN gives it its strength. A Flower Link has a bound on graph size as such that $|N| = \left| T_f \right| + 1$, where the one is from the flower place. State explosion may still occur if concurrency remains, but the FCN with complete restriction on concurrency will bound the number of nodes, | N | = $(|FL|(\left| T_f \right| + 1)) + (2 * \left| T \setminus T_f \right|)$. The second set of terms are derived by the fact that all other transitions in the fully restricted FCN will only have at most one input place and one output place.

5.5.1.3. Global Concurrency

FCN have another interesting potential for showing global concurrency, as seen in Figure 5.12. This FCN has invariant and then concurrent sections. Most of the concurrent sections have become flower links except the ones circled. In this case, the setting of concurrency threshold did not force all branches of the Petri net to be restricted. Instead, the forked process spawns two globally concurrent paths through the FCN before merging near the end of the workflow.

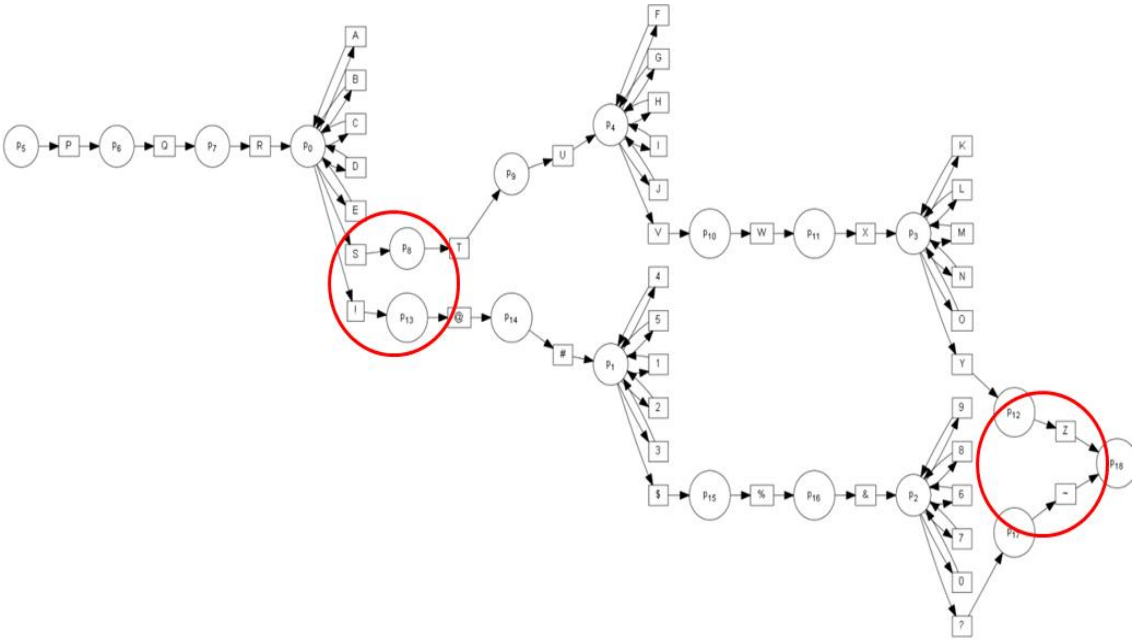

Figure 5.12: Potential for global concurrency maintained in FCN

5.5.2. Behavioral Characteristics of Flower Chain Nets

Behavioral characteristics of Petri net workflow behavior models are based on their ability to replay a log file without over specifying or over generalizing concurrent behavior in the Petri net. The ability to replay log files is captured by the fitness metric, while over-specification is apparent in a high FP Rate. Among over generalized models, the flower model being is most liberal in the patterns of behavior allowed.

Table 5.1 supports the claim that FCN retain positive characteristics of both flower and heuristic models. The experiment to build this table created five datasets with increasing levels of timestamp distortion variance. A modeling algorithm was applied to each data set to discover a Petri net. Then positive and negative instances were replayed through token replay to harvest the source data for these metrics. As the data sets only

contained positive instances, negative instances needed to be generated for the test. A Monte Carlo string generator produced 3684 negative instances by random mutation of activity events by using substitution. The instance generated created instances within a hamming distance of 1-5 of positive instances.

Table 5.1: Behavioral conformance comparison for discovered models

Model	Heuristic			Flower			Flower Chain		
Dataset	FP Rate	FN Rate	Fitness	FP Rate	FN Rate	Fitness	FP Rate	FN Rate	Fitness
CAWF (.001 ms σ^2)	0.000	0.014	1.000	0.000	1.000	1.000	0.000	0.025	1.000
CAWF (.01 ms σ^2)	0.000	0.014	0.999	0.000	1.000	1.000	0.000	0.076	1.000
CAWF (.1 ms σ^2)	0.000	0.036	0.953	0.000	1.000	1.000	0.000	0.253	1.000
CAWF (.5 ms σ^2)	0.386	0.011	0.000	0.000	1.000	1.000	0.000	0.253	1.000
CAWF (1 ms σ^2)	1.000	0.000	0.000	0.000	1.000	1.000	0.000	0.253	1.000

The heuristic model performs well in FP/FN rates until models break under concurrency. Until this point fitness is steadily decreasing. Heuristic models at 0.5 ms σ^2 infer the presence of more states because of timestamp distortion effects. The Flower and Flower Chain Discovery algorithms also detect increasing concurrency, but handle it in different ways. The Flower Model is stable in terms of the metrics, but FN rates are too high for it to be useful. Flower Chains retain positive characteristics of the flower in terms of fitness and false positive rate while also falling between Flower and Heuristic models for FN rates. This result can be improved with CH and CF, which were not used in the test.

5.5.3. Comparison to HMM Approach

The Flower Chain Model is advantageous because it can manage pockets of concurrency and reduce states. This is not the only model that is capable of doing this in process mining, so FCN must be compared against another prevalent method for this purpose: Hidden Markov Models (HMM). HMM are stochastic finite state machines with hidden states and both transition and emission probabilities. In HMM, states, X_n, are hidden and are observed through another set of probabilities $P(a, o)$ where a are the hidden states and o are observations of the states. The probabilities $P(a, o)$ are also called emission probabilities. HMM possess the Markov property that assume that the state of the system at time t+1 depends only on the state of the system at time t, such that

$$\Pr[X_{t+1} = x_{t+1} | X_1 \dots X_t = x_1 \dots x_t] = \Pr[X_{t+1} = x_{t+1} | X_t = x_t]$$

The Markov property is a benefit in HMM because it allows memoryless and state reduced modeling. The number of nodes is bound to the number of nodes when transitions have stationary probabilities and do not have to be duplicated to represent multiple states. HMM used for process mining [42][43] can be effective in detection of improbable transitions in a network, such as ones for semi automated and distributed workflows since events that are never related in a log would have low probabilities for transition.

However, HMM also have disadvantages. Starting with the memoryless property, HMM may be able to detect improbable transitions, but they would not be able to deterministically detect an over abundance of firings within a concurrent region. Another disadvantage is their inability to manage concurrency. Rozinat found that the only way

to express concurrency is by adding states and duplication of transitions [41] or by restricting behavior to 'simple Petri nets'. On the other hand, FCN can range from concurrency (complex Petri net) and non concurrency (simple Petri nets) models.

5.5.4. Evaluation of the Flower Chain Discovery Algorithm

In an event log of size $|L|*|\bar{\sigma}| = n$, steps 1-8 of the algorithm can be done in a single pass ($O(n)$). The complexity of flower candidate selection by scanning footprint matrix is due to the number of transitions $r = |T^2|$. The creation of flower chain models is most limited by the creation of maximal places. There are $O\left(\sum_{0 \leq k \leq r}\binom{r}{k}\right) = O(2^r)$ possible places in a Petri net, but the large majority of these are avoided by composition of compound places from prime places (those with one input and output) and not enumeration. The number of places and transitions in an FCN is strongly tied to the number of flows. A fully restrictive FCN (no concurrency) limits nodes, *N*, and flows, *F*, to a tight limit, subject to $|N| = |P \cup T| = (|FL|(\left| T_f \right| + 1)) + (2 * \left| T \setminus T_f \right|)$ and $|F| = 2 * (|FL|(\left| T_f \right| + 1)) + (2 * \left| T \setminus T_f \right|)$ since every transition in a fully restricted network has two flows by the definition of the flower link.

5.6. Conclusion

We believe that FCN will be helpful in mission situational awareness to provide analysis of third party compliance to exported workflows. FCN also possess abilities to monitor entrance into concurrent regions through checkpoints and monitor behavior through automated or inter organizational sections. The issue of timestamp inaccuracy is one of many potential disintegrations of process mining's core assumption about the accuracy of observation by logging devices. This study has attempted to address these

piecemeal to show the viability of process mining for workflow behavior auditing in mission centric systems and a working methodology for problem identification and development. To solidify this thought, the next section reveals the framework and system for future logging uncertainty testing.

6. LOGGING UNCERTAINTY TESTING

As has been repeatedly shown in this dissertation, process mining lacks approaches to model and conform event data with noise, loss and distortion from system sources but for each issue shown a suitable solution has been found. To identify mission shifts in activity streams inferred from bits on a wire, we developed the workflow shift detector. To automate preprocessing of this data, we developed workflow instance extract and rank algorithms. To enable process discovery and behavioral conformance checking with timestamp inaccuracy, we developed the flower chain Petri net and discovery algorithm.

Yet uncertainty will persist beyond the scope of this study so an evaluation tool that lays the groundwork for future study is required. A general simulation methodology for logging uncertainty testing (LUT) will yield new insights for the core focus of process mining. These insights yield new opportunity for uncertainty modeling, simulation and analysis.

6.1. Logging Uncertainty in Workflow Observation

The idea of ‘logging uncertainty’ was introduced by Rozinat as observation noise [41], but this study expands that definition to include noise and loss from activity inference and timestamp distortion. The basic premise of logging uncertainty testing is that actual behavior and observed behavior vary. In Figure 6.1, a workflow model may specify behavior to a set of human actors. At their discretion, humans choose behavior to achieve mission success, which may or may not be the expected behavior specified by

the workflow model. However, the possibility that events are not recorded as they actually happened exists in some measure on every computer system. By assuming that logging uncertainty does not exist, the process mining community limits itself from expansion into many mission centric environments.

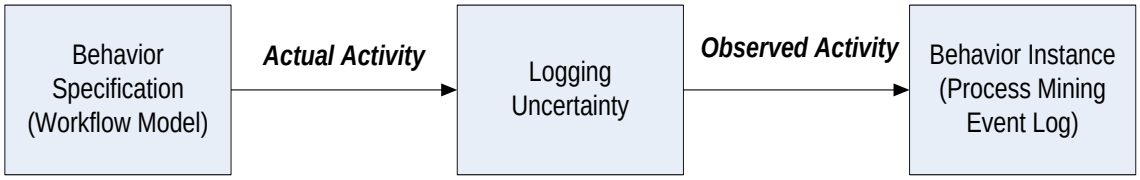

Figure 6.1: Actual and observed behaviors in uncertain environments

Existing solutions like ProM [86] and HMM Experimenter [41] lack the separation between actual and observed behavior. In this section, separation is realized in activity transduction due to stochastic processes. An observation structure of faulty observers is incorporated into the simulation environment. The Workflow Auditing and Simulation Platform (WASP) supports LUT through transduction algorithms for timestamp distortion, noise and loss. WASP also supports a family of repeatable experiments. This is exhibited in a case study for timestamp distortion modeling that uses a pseudorandom exponential function to emulate the effects of queue delays for automated workflows. The section concludes with an evaluation of the WASP against HMM Experimenter and ProM's CPN plug in.

6.1.1. Activity Transduction

Activity transduction is the process by which actual behavior is altered in noisy, lossy and distortion prone event logging environments. This means that through

insertion, deletion and modification of activities, a set of observed behavior instances can be derived from a set of actual behavior instances. Let $\boldsymbol{\alpha}$, $a \in \boldsymbol{\alpha}|\{a_1, a_2, \bullet\bullet\bullet, a_K\}$ be an actual behavior instance in L_A and $\boldsymbol{\beta}$, $b \in \boldsymbol{\beta}$ $\{b_1, b_2, \bullet\bullet\bullet, b_K\}$ be an observed behavior instance in L_O. Then $\beta = \Phi(\boldsymbol{\alpha})$ is a *transduction* of $\boldsymbol{\alpha}$ to β *where* $\Phi(\boldsymbol{\alpha})$ is a transduction system over the input sequence $\boldsymbol{\alpha}$, $a \in \boldsymbol{\alpha}|\{a_1, a_2, \bullet\bullet\bullet, a_K\}$ to produce the output sequence $\boldsymbol{\beta}$, $b \in \boldsymbol{\beta}$ $\{b_1, b_2, \bullet\bullet\bullet, b_K\}$. (Figure 6.2).

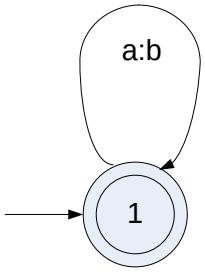

Figure 6.2: Activity transduction

A transduction system is a state machine that produces an output recording from the elements of the input. The simple example in Figure 6.2 records an output *b* from an input *a* so that a string *'aaaa'* would become *'bbbb'.* The simple example is an example of *deterministic distortion,* where a substitution of information has systematically altered the behavior signal that is observed. To allow for a general model for logging uncertainty, a *stochastic distortion* model is required for the transduction system.

To model the transduction $\Phi(\boldsymbol{\alpha})$ for these uses, one can view the observation of an activity as the result of a stochastic process. The observation of an activity can have four basic outcomes.

1) U_A - Actual Activity: The expected activity is observed
2) U_N - Additive Noise: Unexpected activity is observed
3) U_I - Reductive Incompleteness: The expected activity is not observed
4) U_D - Distortion Modification: The expected activity is observed, but it has been updated by a distortive influence in the process (like timestamp modification, attribute modification, etc).

We assume the received behavior signal $\boldsymbol{\beta}$ to be a function of transduction $\Phi(\boldsymbol{\alpha}, U)$of a transmitted signal $\boldsymbol{\alpha}$ and some factor of logging uncertainty, U. U represent an uncertainty factor as a stochastic process and $u \in U$, $\{u_0,u_1,u_2,....,u_n\}$ as instances of the model. With this framework, process analysts can offer models of U for LUT, but a structure for the error prone observations must be designed.

6.1.2. Observation Structure

As is, the ProM framework for CPN simulation presents the observation environment in Figure 6.3. Activity Events are designed to provide the process analyst with the perception that a task is being executed over a period of time. The timing delays described in the previous definition allow for an activity to remain in state s_1^+ during a service delay period. An observer will identify the firing of the transition labeled Act_a (the '*start'* activity) as it transitions between states s_1 and s_1^+. A *'complete'* activity, labeled Act_b is observed by o_2. This is a simple model, but does not provide the researcher with visibility at the *'activity'* level nor does it provide a separation of actual and observed behavior.

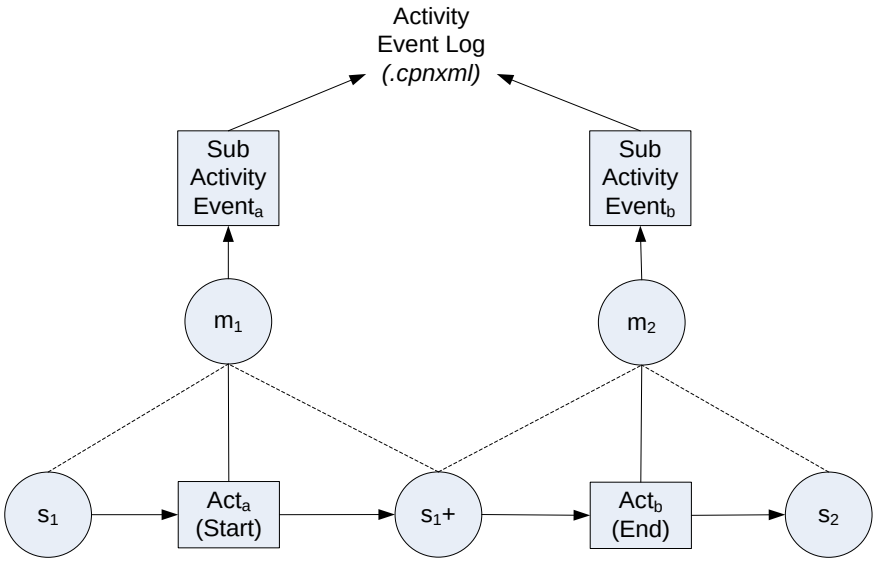

Figure 6.3: Observation structure (ProM)

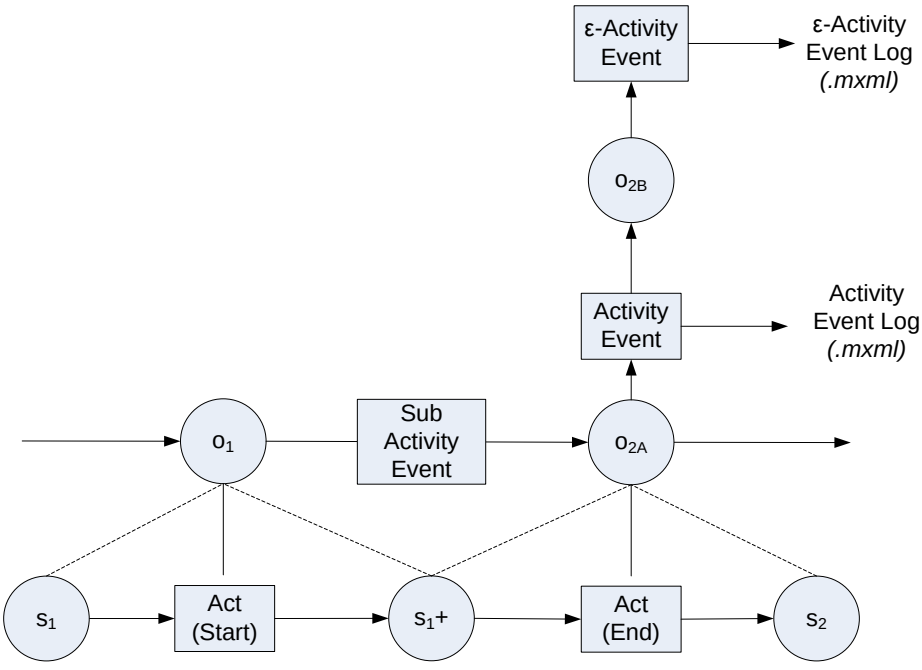

Figure 6.4: Observation structure (proposed)

As opposed to the typical observation structure, this study proposes a layering approach that allows observers to provide multiple outputs. Some observers are directly linked to transitions which they monitor, while others are to linked to logging tools. In Figure 6.4, all observers are connected and share information, $o_1 \rightarrow o_{2A}$ is a reporting event, as is $o_{2A} \rightarrow o_{2B}$. A *data fusion* process occurs at o_{2A} and a *transduction* process occurs at o_{2B}. Data fusion in this case is a trivial aggregation of data fields so it is not discussed further. The real potential of the observation structure is apparent when it runs on top of a simulation environment, described next.

6.1.3. Simulation Environment

Two potential approaches exist for simulating logging uncertainty. The first is the use of a Hidden Markov Model (HMM). Unfortunately, HMM are unable to account for concurrency, so the approach is of limited value for simulation [41]. On the other hand, ProM's CPN plugin uses a Colored Generalized Stochastic Petri Net (CGSPN) which maintains the connection of simulation to the modeling formalism in Workflow Nets. The CGSPN is able to handle more state information and it can represent concurrency, which is critical requirement for workflow simulations.

Even so, some alterations must be made to fit an uncertainty observation structure and activity transduction. The transition observers, $o \in O$, add the potential to observe activities upon the firing of a transition, t. To incorporate them into the simulation framework, we offer the definition for the Colored Observed Generalized Stochastic Petri Nets (COGSPN). The definition does not alter the behavior of a CPN, but rather augments its ability to support LUT by including potentially faulty log writers.

Definition 6.1. (Colored Observed Generalized Stochastic Petri Net) *Let O be a finite, nonempty set of observers, {o_1 , ... , o_n}, a Colored Observed Generalized Stochastic Petri Net (COGSPN) is a 5 Tuple | COGSPN = (CPN, T_1, T_2, W, Φ) where:*

1. *CPN = (P, T, C, I^- , I^+ , M_0) is the underlying Colored Petri net*
2. *$T_1 \subseteq T$ is the set of timed transitions, $T_1 \neq \emptyset$*
3. *$T_2 \subset T$ is the set of intermediate transitions,*
4. *$T_1 \cap T_2 \neq \emptyset$ and $T_1 \cup T_2 = T$*
5. *W = (w_1, ... , $w_{|T|}$) is an array whose i^{th} entry is a function of $C(t_i) \rightarrow \mathbb{R}^+$ such that*

$$\forall c \in C(t_i): w_i(c) \in \mathbb{R}^+ \text{ then}$$

6. *In the case of $t_i \in T_1$ then c_i is a firing delay according to a probability distribution, possibly due to the color $c \in C(t_i)$ or defined stationarily*
7. *In the case of $t_i \in T_2$ then c_i is a firing weight specifying the relative firing frequency due to the color $c \in C(t_i)$*
8. *$\Phi: O \rightarrow T \cup O : |T| \leq |O|$ and $\forall t \in T, \exists o \in O : \Phi(t) = O$*

The definition alters Coloured Generalized Stochastic Petri Nets (CGSPN) [66], where firing delays and weights can incorporate historical information with regards to the service times of a transition. A weighted transition fires in zero service time, while a transition with a firing delay waits during a service period according to an exponentially distributed value. The firing weight of class 2 is used to prioritize the firing of two immediate transitions. The observer function Φ, maps a set of observers to the transitions such that all transitions have at least one observer. The definition allows for multiple observers so that the observations can be streamed to two log files, actual L_A and observed L_O.

6.2. Prototype Implementation of COGSPN

Through modeling and simulation of logging uncertainty, it is possible to measure the effects of uncertainty on process mining methods. Generating a subset of expected behavior instances, one can hope to explore the potential for workflow behavior transduction in the Workflow Auditing and Simulation Platform (WASP), which contains a suite of transduction, process discovery and conformance checking algorithms where uncertainty monitoring experiments the WASP are repeatable.

6.2.1. The WASP Testbed

The WASP (Workflow Auditing & Simulation Platform) is a tool that a process analyst can use to discover and analyze the control flow perspective of workflows as they are captured in event logs. The WASP's unique analytical components are specifically geared to understanding the effects of various factors of uncertainty on process discovery and conformance checking.

The WASP can be separated into two major components, data generation and data analysis. Figure 6.5 shows the data generation component. The workflow model designer (upper left) is the component of the WASP that creates workflow models. It achieves this through YAWL (*Yet Another Workflow Language*) [70], which is an open source business process design program. One can employ some of the inherent analysis and design capabilities of this tool. Actual behavior is generated by a discrete event generator (lower left of Figure 6.5). The discrete event generator needed to be capable of stochastic timing and path choices. CPN Tools [65][87], has a history of supporting process mining endeavors of this sort [85][88].

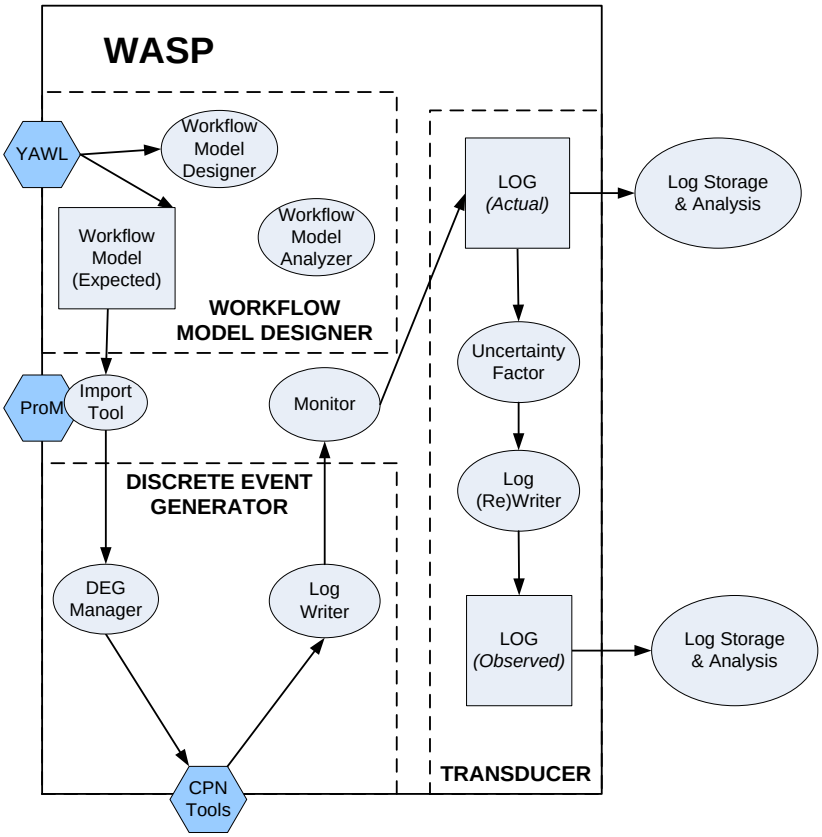

Figure 6.5: WASP simulation architecture

To generate activities as discrete events in a log, CPN Tools requires a petri net representation and acceptable file format. ProM provides this, but also gives the capability to generate distributions for choice for the transitions and timing in the workflow model. These transitions emulate human discretion within the expected behavior of the workflow model. A DEG Manager can automate the program simulation from Java [87] and writes events to a log, so long as the CPN has been instrumented to do this. The result of DEG is actual behavior.

The log with the actual activities is stored or imported into the WASP Transducer (right of Figure 6.5). The Transducer accepts an activity event in a log as input and writes a transduction of the activity event to another log as output. The nature of the transduction is due to the uncertainty factor, $u \in U$. The resulting set of behavior instances at the output is observed behavior. Both sets of data, Actual and Observed are presented for analysis.

The data analysis component is shown in Figure 6.6. Data analysis occurs at three levels, basic analysis, process discovery and conformance, which are arranged hierarchically on the right of the figure. Either actual or observed behaviors can be analyzed, but often both are analyzed together to evaluate differences in the logs. In basic analysis, logs are scoured for activity frequencies, timing delays, and basic bigram structure. Bigram structure is the set of any two adjacent activities in the log. Basic analysis is important for initial understanding of uncertainty in the logs, but process discovery is necessary for higher understanding. In the process discovery phase, behavior instances are clustered and mined to produce various perspectives on behavior. In the conformance phase, token replay of the events in both logs can be replayed on discovered processes and their low level rules compared among expected, actual and observable behavior. Conformance Checking yields insights about the usefulness of models in applications. All three levels yield artifacts, metrics and insights about the nature of uncertainty. For comparison of algorithms implemented in WASP and in ProM or other process miners, logs may be ported to these other systems and analyzed separately.

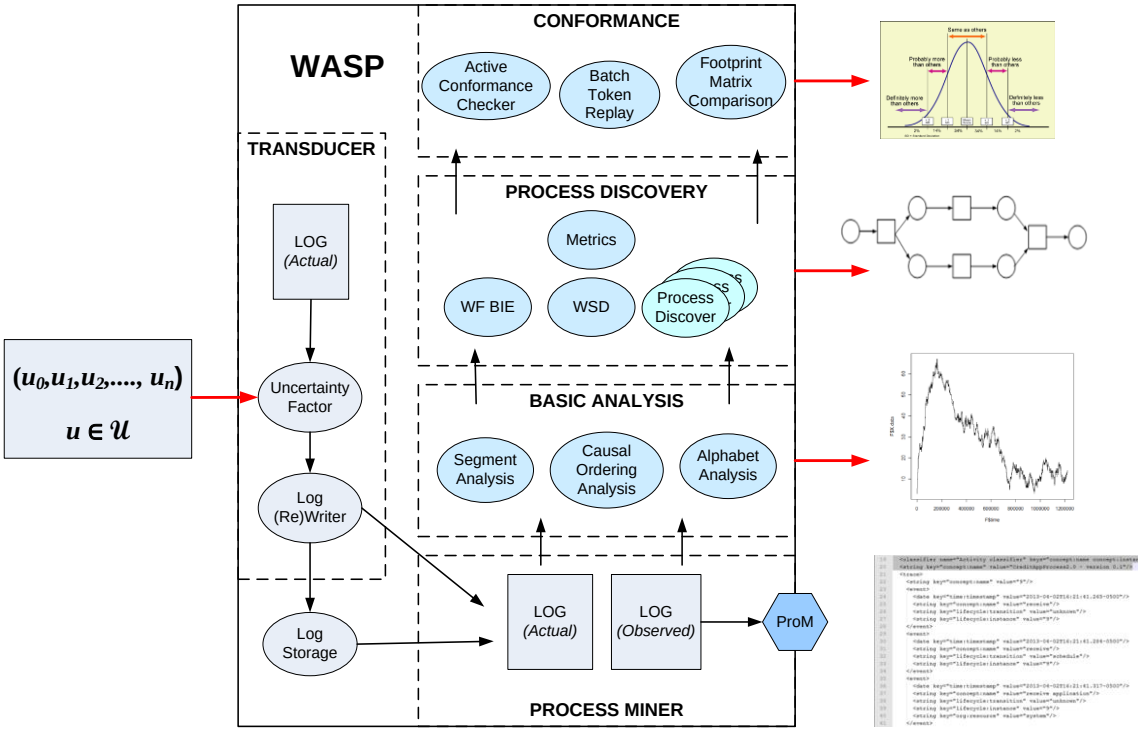

Figure 6.6: WASP analysis architecture

6.2.2. Experiments for Logging Uncertainty Testing

In order to observe the impact of uncertainty, one would need to place an observer between the behavior specification and behavior instance and log, with full view of the inputs and outputs of the channel. LUT should be done for a variety of purposes such as scenario detection experiments, which led to the workflow shift detection problem, uncertainty factor analysis which led to the need for WIC extraction and ranking or discovery experiments that yield new models (e.g. FCN) and discovery algorithms (FCDA).

6.2.2.1. Scenario Detection and Evaluation

Finally, the user of the WASP can import a data set or manually craft scenarios that can be included in a test set for detection by instance replay tools. They work by replaying the log over a discovered model in a strict or approximate fashion. Suitable metrics would include the pass/fail ability of the log to replay the scenario or identify breaches of expected behavior. Approximation techniques can also help a scenario detection experiment to be evaluated on a gradient between pass and fail. Broadly, scenario detection experiments test the process mining tool chain to detect fraud, information leaks, systemic policy breaches, performance bottlenecks or a shift in mission state/operations intensity as described in Section 3.

6.2.2.2. Uncertainty Factor Analysis

On the left of Figure 6.6, the modeling capability of the WASP is showcased. Uncertainty modeling is done in the next section, but for now it is important to point out the ability of the WASP to incorporate various modeling approaches in an iterative fashion, which allows millions of trial runs to alter sequences in logs under vaious parameters. Uncertainty factor experiments transform an event log in the test case while maintaining a control case.

Analysis from the test and control logs may determine how a single algorithm performs in the presence of uncertainty. Algorithms can be shown to be robust to an uncertainty factor by evaluating artifacts produced at various analysis phases of process mining: basic, discovery and conformance. In general, three possibilities can occur with comparison at these phases. First, a method may be considered to be robust if the

comparison tools show no significant deviation between mined artifacts. Second, comparison tools show unacceptable deviation between test and control to show the method is not robust. Third, partial deviation between artifacts produced from the test and control logs indicates an area which algorithms may perform well sometimes and not others. In the third case, suitable techniques may be applied to preprocess the data, correct the algorithm to be less susceptible to logging uncertainty as shown in Section 4 and Section 5.

6.2.2.3. Discovery Model Conformance

In discovery experiments, an logging uncertainty is controlled while multiple algorithms are used to discover and compare process artifacts. The use of pseudorandom number generators in creating the log files provides a consistent transformation of the log so that the algorithmic inputs do not vary between methods. In addition, discovery and conformance experiments may be performed over ranges of workflow models, seeds and uncertainty models to allow for evaluating algorithms.

In modeling from sets of workflow instances, three possibilities can occur with discovery between two algorithms. First, both algorithms can produce the same process model or within an insignificant margin as measured by a metric. In the absence of a divergent metric, these algorithms are said to perform equivalently for the given uncertainty factor. Second, resulting models may be completely different, meaning that the artifacts they produce do not conform to one another. In this case, it will be important to use some metric to determine which is better or if neither perform well enough. The third possibility is that regions of the discovered process model are the

same and others are not and the difference between the discovered model is significant by some metric. The quality of the metric will help the process analyst choose between algorithms as before.

In summary, discovery experiments can justify the use of one algorithm versus another. With suitable and expressive metrics, they can help the process analyst make tradeoffs in generalization, algorithmic performance and behavioral conformance as described in Section 5.

6.3. Structural Conformance with Timestamp Distortion

The WASP has structural conformance capabilities as well as other evaluative tools to evaluate process mining with logging uncertainty. The use of pseudorandom number generation makes experiments repeatable if needed. As a case study for an uncertainty factor experiment, timestamp distortion analysis is continued from the previous section by exploring timestamp distortion using an exponential delay model to simulate the effects of varying service delays. To analyze the effects, *structural* conformance is monitored in discovered Petri nets.

6.3.1. Footprint Matrix Conformance

Structural conformance [89] refers to the analysis of nodes and flows in a Petri net. Footprint conformance is a specific approach for structural conformance that analyzes the rules that a discovery algorithm produces to create a Petri net. The running example in section 4 is used as the source data for structural conformance testing. Further, an exponential timestamp distortion technique is employed to distort the timestamps in a test log.

The footprint matrix from the test log is compared to the footprint matrix of an oracle log. An interim step in process discovery using the algorithm is the creation of a footprint matrix. Footprints can be compared to investigate the changes in causal dependency in the presence of timestamp distortion. Recalling the earlier discussion of rules inference by exposing log ordering relationships, the two matrices in Figure 6.7 are to be compared (with an oracle (Top) and with a timestamp distorted file (Bottom)).

Footprint Matrix (Oracle)												
Activity	A	B	C	D	E	F	G	H	I	J	K	L
A	#	->	#	#	#	#	#	#	#	#	#	#
B	<-	#	||	->	#	#	#	#	#	#	#	#
C	#	||	#	#	#	#	#	#	#	#	#	#
D	#	<-	#	#	->	#	#	#	#	#	#	->
E	#	#	#	<-	#	->	#	#	#	#	#	#
F	#	#	#	#	<-	#	->	#	#	#	->	<-
G	#	#	#	#	#	<-	#	->	->	#	#	#
H	#	#	#	#	#	#	<-	#	||	->	#	#
I	#	#	#	#	#	#	<-	||	#	->	#	#
J	#	#	#	#	#	#	#	<-	<-	#	#	#
K	#	#	#	#	#	<-	#	#	#	#	#	#
L	#	#	#	<-	#	->	#	#	#	#	#	#

Footprint Matrix (Test)												
Activity	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**
A	#	#	#	#	#	#	#	#	#	#	#	#
B	#	#	->	->	#	#	#	#	#	#	#	#
C	<-	#	->	->	->	#	#	#	#	#	#	#
D	#	#	<-	<-	#	->	#	#	#	#	#	#
E	<-	#	<-	#	->	->	#	#	#	#	#	||
F	#	#	#	<-	<-	#	->	->	->	#	->	<-
G	#	#	#	#	#	<-	||	||	#	->	#	#
H	#	#	#	#	#	<-	#	||	||	->	#	#
I	#	#	#	#	#	<-	||	#	||	->	#	#
J	#	#	#	#	#	#	<-	<-	<-	#	#	#
K	#	#	#	#	#	<-	#	#	#	#	#	#
L	#	#	#	||	#	->	#	#	#	#	#	#

Figure 6.7: Footprint for oracle (top) and test (bottom) timestamp distortion

Footprint Matrix (Difference)												
Activity	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**
A		->:#	#:->	#:->								
B	<-:#		||:#	->:#								
C	#:<-	||:#	#:->	#:->	#:->							
D	#:<-	<-:#	#:<-			#:->						->:||
E			#:<-									
F				#:<-				#:->	#:->			
G								->:||	->:||	#:->		
H						#:<-	<-:||					
I						#:<-	<-:||					
J							#:<-					
K												
L				<-:||							

Figure 6.8: Difference matrix of footprint rules

The test log was developed when the average service time was 200 ms. By visual inspection of the footprint matrices, one can perceive the differences between the two logs as a general increase in the number of rules (# → {→, ← or ||} and an increase in concurrency relations ||. This is to be expected because of the effect of reordering. In Figure 6.8, the differences are identified by removing rules that are equivalent.

Visual inspection does not quantify differences, so footprint conformance is a way to quantify the differences between rules in process models. The simple metric sums the number of different rules over the total, including # relations and subtracts the fractional number from 1. In this case, $1 - \frac{29}{144} = 0.7986$. By measuring footprint conformance, one can graphically summarize the effect of timestamp distortion on a process discovery algorithm. To set a lower bound for footprint conformance using the naïve α-algorithm, the WASP iterates on increasing values of timestamp distortion. The result can be seen in Figure 6.9.

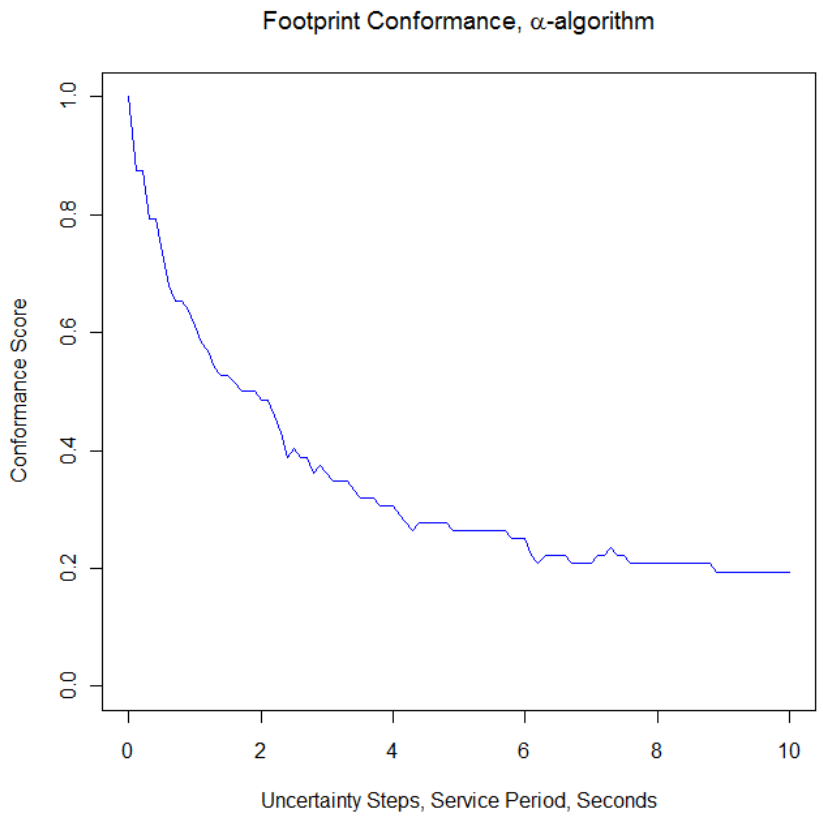

Figure 6.9: Degrading footprint conformance scores

The function exhibited by the footprint conformance metric can be strongly correlated to a precision metric for bigram retrieval, Figure 6.10. Bigram precision is calculated using $\frac{|\, b \in G(L_O) \cup b \in G(L_D)\,|}{|\, b \in G(L_O)\,|}$ to measure the proportion of relevant causal ordering relationships in an event log to all bigrams, *b*, in the log. The exponential decay exhibited in both graphs implies a correlation between the qualities. Therefore, improving the precision of the bigrams in the log it may also have an effect on conformance as well.

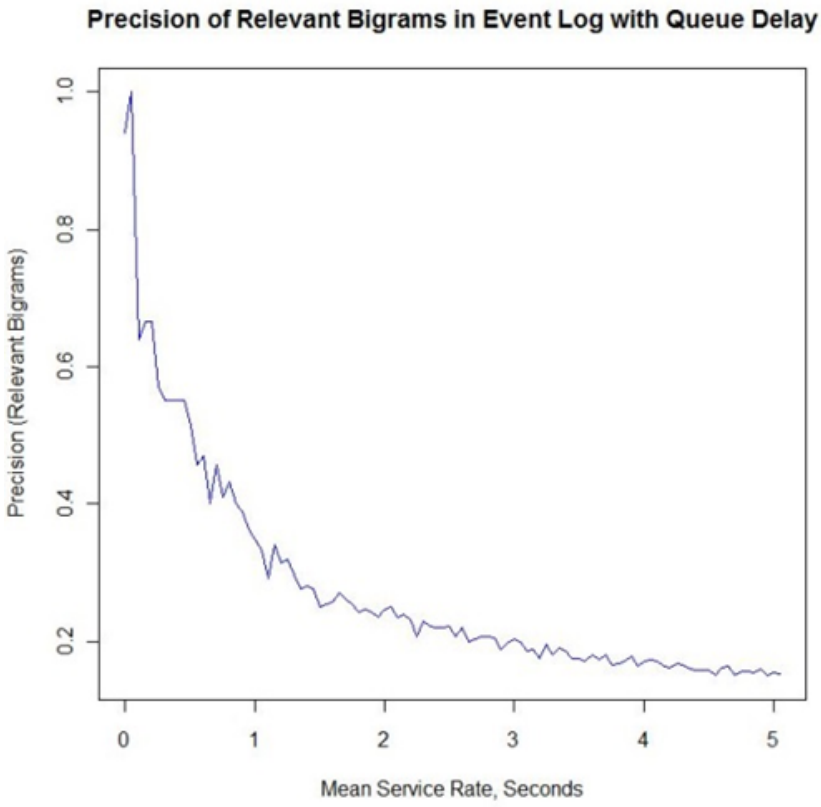

Figure 6.10: Bigram precision with increasing distortion

6.3.2. Heuristic Approach and Iterated Testing

To capitalize on this, well supported rules can be distinguished to separate stronger causal relationships from weaker ones. The heuristic approach to process discovery is based on a statistical counting method of the log ordering relations. The interested reader should refer to [49] for a further discussion on the method, but the basic process involves counting the number of ordering relations that occur for evidence of their support. Then, a dependency measure is used to identify the strength (from -1 to 1), of the log ordering relation. The dependency measure is defined as:

$$a \Longrightarrow_L b = \begin{cases} \dfrac{|a >_L b| - |b >_L a|}{|a >_L b| - |b >_L a| + 1} & if\ a \neq b \\ \dfrac{|a >_L a|}{|a >_L a| + 1} & if\ a = b \end{cases}$$

The dependency measure is applied against the causal counting matrix producing the dependency matrix seen in Figure 6.11. The resulting values indicate the strength of ordering relations based on their statistical prevalence.

Dependency Matrix												
Activity	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**
A	0	0.667	0.975	0.966	0	0	0	0	0	0	0	0
B	-0.667	0	-0.042	0.997	0	0	0	0	0	0	0	0
C	-0.975	0.042	0	0	0	0	0	0	0	0	0	0
D	-0.966	-0.997	0	0	0.997	0	0	0	0	0	0	0.947
E	0	0	0	-0.997	0	0.997	0	0	0	0	0	0
F	0	0	0	0	-0.997	0	0.978	0.977	0.991	0	0.995	-0.947
G	0	0	0	0	0	-0.978	0	-0.214	-0.229	0.951	0	0
H	0	0	0	0	0	-0.977	0.214	0	-0.418	0.987	0	0
I	0	0	0	0	0	-0.991	0.229	0.418	0	0.957	0	0
J	0	0	0	0	0	0	-0.951	-0.987	-0.957	0	0	0
K	0	0	0	0	0	-0.995	0	0	0	0	0	0
L	0	0	0	-0.947	0	0.947	0	0	0	0	0	0

Figure 6.11: Dependency matrix for heuristic mining algorithm

Figure 6.12 shows repeated trials of the footprint matrix comparison for various levels of the heuristic. The region contained by the upper boundary of the most stringent heuristic based approach and the lower boundary of the pure approach indicates the range of potential solutions for model conformance with timestamp distortion using heuristics based methods. This serves as a guide for potential solutions and should also bound expectations for process models emerging in environments with logging uncertainty.

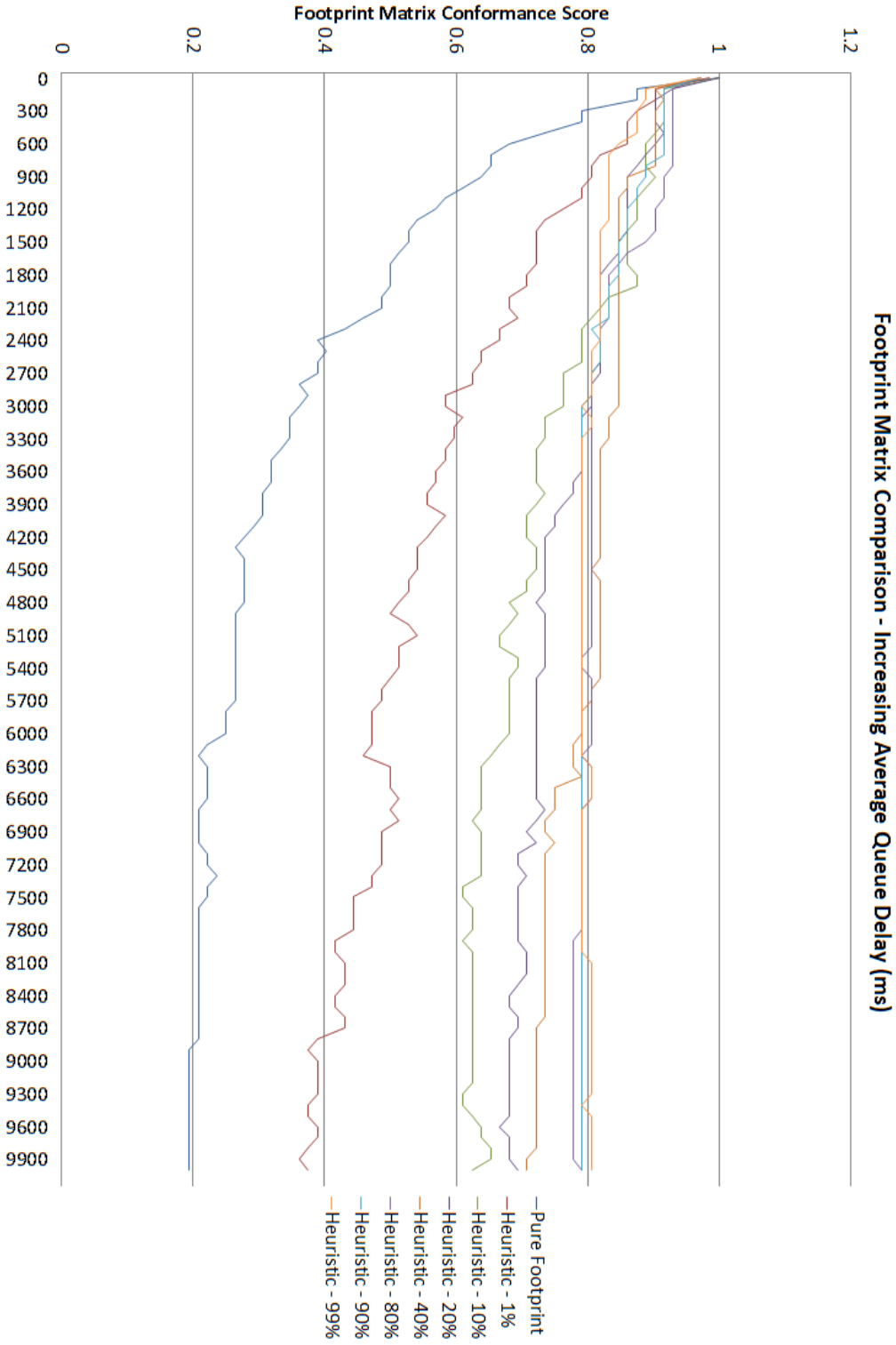

Figure 6.12: Degrading structural conformance from increasing queue delay

6.4. Testbed Comparison

Three systems stand out as appropriate comparisons to the WASP. ProM incorporates two plug-ins, the CPN simulator and the HMM experimenter. Both offer a process analyst an ability to simulate workflow nets. WASP is built on tools that ProM uses. The HMM experimenter lacks some functionality because the HMM that form the basis of simulation are limited in concurrency. Our extension of the simulation environment is the idea of split observation. The WASP offers all categories of capabilities as is seen in ProM's CPN and HMM experimenter, yet because of its immaturity and lack of production level quality, the WASP lacks the analysis power of its counterparts. The summary of these characteristics is seen in Table 6.1.

Yet the WASP is unique in its ability to support LUT. The HMM Experimenter considered the idea of logging uncertainty, but did not describe it. Our tool offers a number of algorithms above ProM based tools, primarily in the distortion category. Noise is a common feature in these tools, but noise padding, which helps to simulate effects seen in noisy, continuous log files is also unique. A summary of these uncertainty testing capabilities is in Table 6.1 and Table 6.2

Table 6.1: Evaluation of simulation capabilities in WASP and ProM tools

Simulation Capabilities for LUT			
Feature	**ProM - CPN**	**ProM - HMM**	**WASP**
Bank of Algorithms	✓	✓	✓
Complex PN	✓		✓
Stochastic Rate	✓		✓
Stochastic Delay	✓		✓
Prob. Transition	✓	✓	✓
Split Observer			✓

Table 6.2: Evaluation of modeling capabilities in WASP and ProM tools

Modeling Capabilities for LUT			
Feature	**ProM - CPN**	**ProM - HMM**	**WASP**
Distortion			✓
Lossiness	✓		✓
Queue Delay			✓
Clock Drift			✓
Noise (Extraprocess)	✓	✓	✓
Noise (Intraprocess)	✓	✓	✓
Noise Padding			✓

6.5. Conclusion

This section described how a testbed and methodology for LUT was defined and implemented. The system transforms behavior instances and observes the impact of uncertainty by measuring the difference between process models derived from L_A and L_O. The WASP makes it is possible to model, analyze and overcome the effects of logging uncertainty.

Mission centric collaboration requires modern auditing methods for workflow behavior auditing. In this study, we have sought our approach to modern auditing through process mining methods. However, the existence of noise from activity inference and system errors is problematic for process mining because they effect the perception of causal relationships. To address these concerns, we have developed algorithms, process models and a methodology that carves a path for process mining with logging uncertainty. With the WASP, we have developed the solutions towards the goal of realizing workflow behavior auditing for mission centric collaboration.

7. CONCLUSIONS AND FUTURE WORK*

To advance mission centric collaboration with tools for situation awareness, this study has investigated workflow behavior auditing in a battle rhythm driven computing environment. In particular, the major emphasis of this thesis is to show the feasibility of process mining for environments with dynamic mission states and logging uncertainty.

Modern mission centric collaboration presents technical challenges in the form of identification of dynamic mission state changes, extraction of workflow instances from noisy, unstructured data and mitigation of distributed system effects on process discovery algorithms. To this end, several research questions were posed to validate the application of process mining. First, are workflow changes in a battle rhythm detectable

---

*Part of the data reported in this section is reprinted with permission from "APSAT: A Framework for Modeling and Analysis of Workflow Dynamics in Mission Centric Systems." by J. Pecarina and J.C. Liu, 2012 in *Proceedings of the 2012 Conference on Collaboration Technologies and Systems*, Westminster, Colorado, 2012, pp. 575–582, Copyright [2012] by IEEE.

*Part of the data reported in this section is reprinted with permission from "SAPPHIRE: Anonymity for Enhanced Control and Private Collaboration in Healthcare Clouds." by J. Pecarina, S. Pu, and J.C. Liu, 2012 in *Proceedings of the 4th International Conference on Cloud Computing Technology and Science*, Taipei, Taiwan, 2012, pp. 99–106, Copyright [2012] by IEEE.

*Part of the data reported in this section is reprinted with permission from "Behavior Instance Extraction for Risk Aware Control in Mission Centric Systems." by J. Pecarina and J.C. Liu, 2013 in *Proceedings of the 3rd International Conference on Cognitive Methods in Situation Awareness and Decision Support*, San Diego, California, 2013. pp. 45-50, Copyright [2013] by IEEE.

from activities inferred from packet traces? Second, can workflow instances be extracted from activities inferred from packet traces to provide source data for process mining? Third, is Petri net discovery still structurally and behaviorally appropriate when causal dependency is undermined by timestamp distortion? Fourth, is long term testing and development of process mining in environments with logging uncertainty supportable?

We believe that by addressing these challenges, workflow behavior auditing will help decision makers achieve mission success by managing collaborating forces and systems with a mission centric mindset.

7.1. Summary of Research Findings

The work in this dissertation answered the research questions by defining technical problems and developing algorithmic solutions that address limitations in the current state of the art in process mining. The major contributions of this study include WSDA to find Mission State Changes in a battle rhythm, WIC-Extract to provide source data for process discovery/conformance of workflow behaviors, FCN/FCDA to enable service to service and inter-organizational auditing with timestamp uncertainty and the WASP, which is a testbed for evaluation and development of logging uncertainty testing of process mining methods for workflow behavior auditing.

WSDA is a statistical change point detection algorithm that detects concept drift in noisy event logs. WSDA uses a sliding window scheme to capture statistical frequencies of general features of resource access actions. WSDA found workflow shifts in activities inferred from packet traces, detected workflow shifts will make Petri nets more concise and used an online approach to finding change points in workflow event

data. We believe it may also be used to reduce extraction requirements for Workflow Instance Collection by splitting logs into workflow periods, where particular behaviors may have a higher rate of occurrence.

WIC-Extract is a string processing algorithm that finds approximate repeating subsequences in a symbolic string that represents actions in an event log. Without specific knowledge of the targeted activity symbols, WIC Extract and Rank algorithms provide a tool for process analysts to extract approximately repeating activity sequences. Our workflow behavior features capture inherent characteristics of workflow sequences to improve the ranking of sequences to an analyst. The algorithm also features a genetic algorithm that configures the weights in the ranking function so an analyst can select from top results. WIC Extract found workflow instances in activities inferred from packet traces of simulated battle rhythm data. The algorithm provides correct output, returning all WIC within the analyzed log segment and bounds the noise in the output event logs using the tunable parameter. We believe that this tool will be beneficial to the process mining community to expand data sources into network traffic and other unstructured sequential data.

FCDA is a process discovery algorithm that accepts a potentially timestamp inaccurate event log and produces a Petri net model to represent concurrency bounded behavior from the log. Our solution showcases a stepwise scaling approach to allow more concurrent structure into the model than Flower Petri nets. At the same time, it features a minimal footprint while reducing the false positive rate of heuristic algorithms. The solution can be incorporated with anomaly detectors using its structure

to carry historical and organizational data. We believe this model could be scaled to incorporate more expressive behavior within a threshold of minimum false positives. We also believe that the model will be useful in managing semi-automated and distributed workflows, where concurrency from timestamp inaccuracy is expected to be high to enable service to service and inter-organizational transaction auditing in timestamp uncertain scenarios for cloud workflows.

Finally, the WASP is a testbed to evaluate and develop process mining methods in logging uncertain environments. The WASP possesses a simulation framework that observes and transforms activities generated by a colored generalized stochastic Petri net. We demonstrated repeated trials of Logging Uncertainty Testing (LUT), answering a critical need in the field of process mining to evaluate the robustness of process mining techniques in the presence of timestamp distortion, observation noise and loss. The methodology is demonstrated throughout our study as empirical evidence of repeatable modeling, analysis and algorithmic design approaches. We believe this tool will be vital to advance process mining technology for mission centric collaboration.

From this empirical evidence, it is a reasonable claim that process mining can be applied with dynamic mission states and logging uncertainty. Further, the development of an evaluative framework for process mining with logging uncertainty ensures future analysis and development in this new focus area.

7.2. Implications for Mission Situational Awareness

The empirical evidence suggests the viability of process mining for mission situational awareness. The underlying premise for this claim is that workflow execution

contributes to the organizational mission. Given the premise, since we have shown process mining to be feasible in the battle rhythm environment, we believe that mission situational awareness with workflow behavior auditing under the process mining framework is also feasible by extension.

Workflow execution in an air operations center supports the intermediate milestones in the air battle rhythm. Repeated and successful workflow executions contribute to the production of an ATO, which is the overall mission of the organization. Workflow execution is captured in event logs as activities and workflow instances. Event logs can be audited to assess and optimize mission centric collaboration. Therefore workflow behavior auditing, defined as functions that "assess and optimize workflow performance" [13], evaluates the supporting elements of the organizational mission.

It follows that if workflow behavior auditing can provide conformance checking and process discovery of the supporting elements of a mission, then mission situational awareness is possible through workflow behavior auditing. Our assertion is that the critical path to mission situational awareness through workflow behavior auditing is realized in process mining. Process mining can discover a workflow behavior model, check the conformance of observed behavior to a given workflow model, or enhance a model with situational awareness [18]. Decision makers and process analysts have used process mining for resource allocation [85], intrusion and fraud detection [52][53], forensic security [54][55][56] and access control [57][58][59][60]. Thus, process mining has been shown to be effective for workflow behavior auditing.

Nevertheless, we must characterize assumptions as those for which this theoretical implication will hold. We recognize that our result was obtained in a simulated environment where noise, loss and distortion were still somewhat limited. We also recognize that in order to detect workflow shifts and extract workflow instances, event data needs to record approximately repeating sequences of activities. The statistical prevalence of workflow behavior is likely when workflow behavior is repeated more often than noise. This is important to support a final assumption that causal dependency could still be inferred despite the existence of logging uncertainty.

7.3. Lessons Learned and Future Work

The theoretical implication raises critical questions about process mining for mission situational awareness. Questions concerning the validity of simulated workflow data and simulated uncertainty models must be addressed to maximize the applicability of this study for broad use. Questions concerning real world implementations and applications of this work must also be addressed to advance workflow behavior auditing in the mission centric environment. By exposing the limitations of the current study, we identify several avenues of future work.

7.3.1. Data Sources

Our foremost limitation is the availability of realistic data, leading to three lessons learned. First, data was simulated for our study by scenarios of potential workflows in sterile and uncertainty controlled environments. While we justify the controlled simulation of workflow behavior data as a means to isolate the potential effects of uncertainty and dynamism on process mining algorithms, widespread adoption

of our algorithms can only come if they are tested using real world data sets. Second, RAA harvesting explored a relatively small region of the unstructured data in enterprise networks. The parsing activity for the SMB protocol would need to be repeated on other protocols and systems to infer activities across the enterprise. Third, enterprise networks generate a lot of data, leading to a concern that the methods employed for data preprocessing in this study may not scale well. Thus, data sources should also be large scale, requiring scalable methods for data preprocessing.

To address the first lesson learned, we believe the value of process analysis will increase in the eyes of mission stakeholders so that there will be opportunities to harvest new data sources under more realistic conditions. A significant technical hindrance here is privacy, so techniques to anonymize or audit privately or confidentially could encourage organizations to share workflow behavior datasets. The second lesson learned of expanded activity inference algorithms must follow two paths. First, there are other protocols and systems to analyze. For example, web based process mining should be considered for online collaboration environments. Second, non deterministic techniques for activity inference should be considered. In our work, human activity is inferred from protocol and system events using rule based parsing techniques, but future work could consider probabilistic inference.

Finally, the scalability concern for data preprocessing demands future works to monitor and preprocess enterprise networks with modular and parallel systems, as shown in Figure 7.1. Data harvesting, (Actionizer in the figure) could be implemented by placing observation points in networks at bottlenecks (like portals or placed near the

hosts being monitored). Harvesters would output data streams like RAA streams, which could be segmented by workflow shift detection. WIC extraction could be executed over workflow periods in parallel by accessing and dividing preprocessing tasks from a common data store. Modeling tools could operate concurrently over data maintained by process analysts.

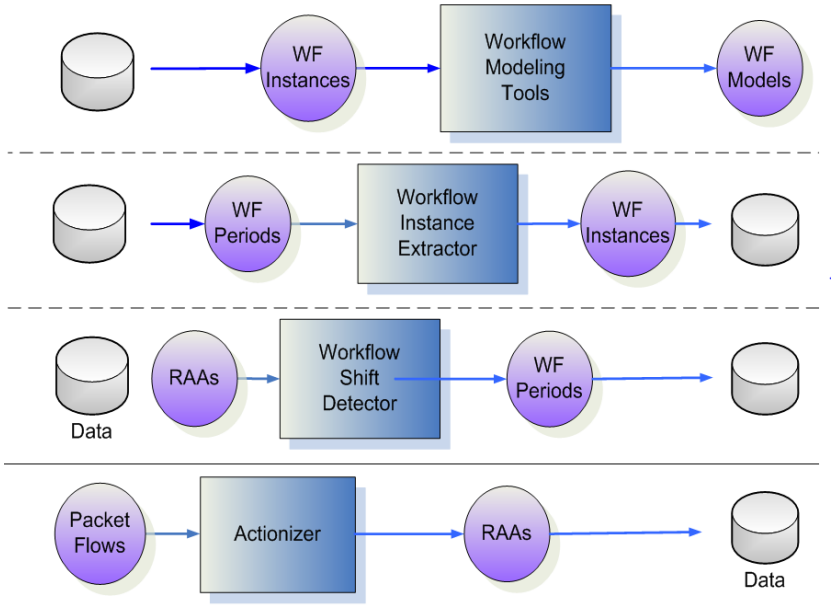

Figure 7.1: Modular and parallelized data preprocessing

7.3.2. Logging Uncertainty Testing

Algorithms for temporal uncertainty, noise padding and lossy observation in this dissertation were initial models for logging uncertainty. Other sources of logging uncertainty remain to be explored, such as log manipulation and logging queue overflows to name a few. Improvements or domain specific alteration of logging uncertainty models may help auditors gain further insight into process mining with

faults. As for modeling uncertainty, future work should also establish benchmarks and guidelines for LUT. This has been difficult to this point because of the lack of benchmarks in process mining in general, but as new data sets and evaluative frameworks emerge, there will be opportunities for LUT standardization as well. This may help lead to establishing theoretical bounds for conformance between observed workflow behaviors to an oracle model given a level of uncertainty.

Our evaluation testbed has expansion potential as well. It is true that ProM is a de facto standard for process mining experiments, but the community that uses it has a splintered focus. We intend to extend capabilities of the WASP rather than incorporating it as a plug-in to ProM to give the WASP a unique feel for its core competency in logging uncertainty. The prototype definitely needs better algorithms, an XES reader and banks of workflow nets to simulate from.

7.3.3. Workflow Behavior Auditing Applications

This dissertation featured components of the Access Profile Statistical Analysis Tool (APSAT) [68], which was a prototype to harvest, extract and model workflow behaviors amongst distributed file systems. Plans to expand the APSAT framework, shown in Figure 7.2, should consider scalability and parallel execution. They should also consider practical ways to measure mission relevance and health and methods and models for conformance. The tool was limited in scope and reach, but it exhibited the basic necessity of modern mission centric auditing, a stepwise approach to building knowledge about worker behaviors and how they compare to a baseline mission. Future iterations of the framework may lead to advanced workflow behavior applications.

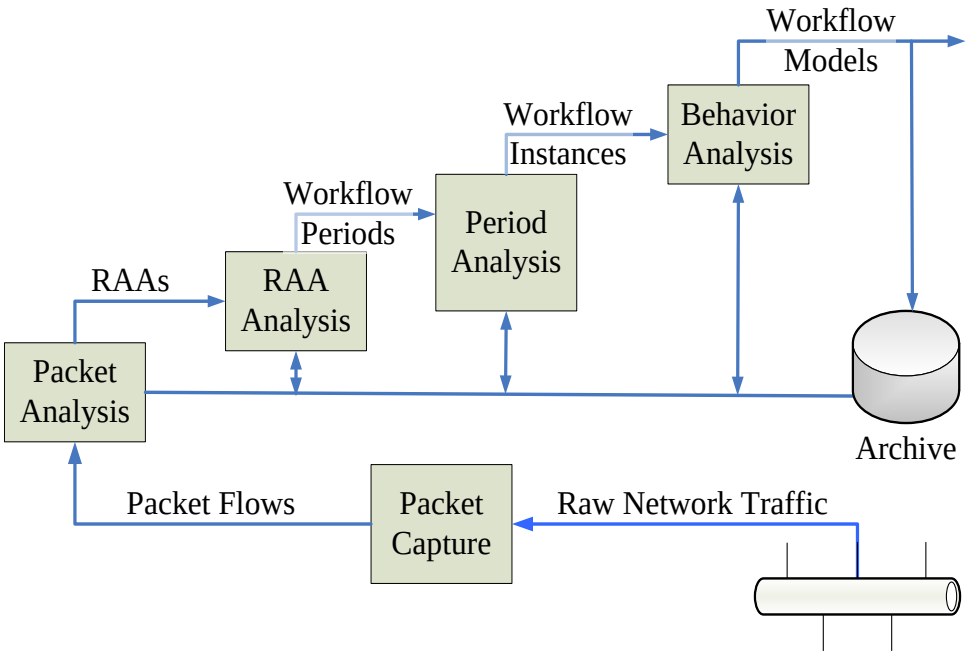

Figure 7.2: APSAT prototype auditing framework

Along these lines, a mission situational awareness could be a valuable commodity for access control, authentication and network management. A corporate workflow profile, coupled with an appropriate risk model may be used to assign a level of trust between a resource owner and requestor. Decision makers would understand more about the health of their organization to make network policy decisions with the profile as justification. Initial system prototypes are needed to demonstrate the value of assured and adaptive systems to a broad base of researchers and mission owners.

Policy and privacy in next generation mission centric systems will depend on advances in auditing and applications that can utilize adaptive behavior discovery and conformance tools. Two platforms that rely on modernized workflow behavior auditing illustrate a long term vision for assured and adaptive systems. The first is the

Releasability Gateway, which uses mission relevance as an attribute for authentication and fine grained access control. The second is the Risk Aware, Mission Parameterized (RAMP) policy framework [81] that senses workflow dynamics to recommend and adapt security policy according to mission goals.

7.3.3.1. Releasability Gateways

Releasability Gateways address data privacy through dynamic trust and trust management [90][91][92]. Figure 7.3 shows the conceptual distributed architecture using Releasability Gateways to form a layer of abstraction for a Health Information Exchange (HIE) [93] that provides dynamic access to third party organizations. As a proxy or access portal, HIE could redirect static mechanisms in Kerberos [94] and Role Based Access Control (RBAC) [95] for dynamic attributes. Bootstrapping a Kerberos-like protocol to supplant identity management with trust verifications [96] allows policy authorization to proceed from the data owner while security enforcement mechanisms remain with the data storage location. Workflow auditing can yield a mission relevance attribute to be employed within trust negotiation [97][98] or Attribute Based Encryption [99][100] for portal authentication. For mission relevant authorization to data, Attribute Based Access Control [101], usage control [102] or workflow based access [103] could provide dynamic and fine grained access for semi-trusted third parties. Mission relevance would be captured as a worker's positive contribution to a workflow. Dynamic access can be regulated through a credential management tool on the gateway that changes access privileges to conform workflows to mission profiles.

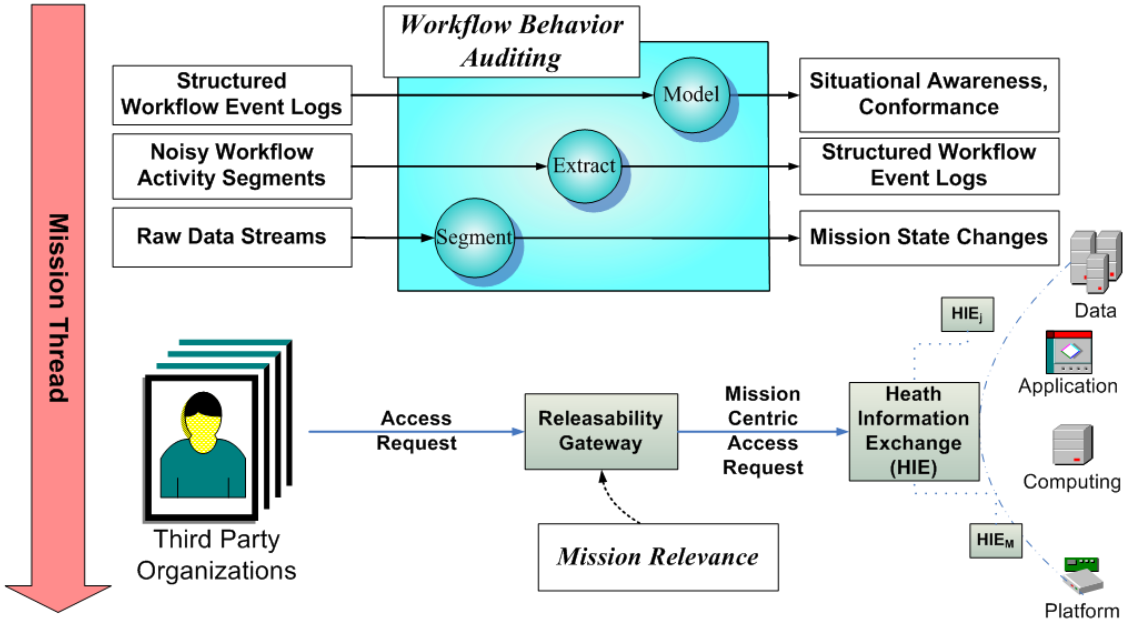

Figure 7.3: Releasability Gateway for dynamic access and authentication

7.3.3.2. Risk Aware, Mission Parameterized Policy

The Risk Aware, Mission Parameterized (RAMP) policy framework [81] (Figure 7.4) is designed to adapt security policy based on the perception of mission state. The proposed framework calls for log analysis tools [104][105][68] to capture raw sequential data, segmenting it into workflow activity sequences. RAMP retrieves workflow instances from segments of interest using string processing and information retrieval techniques to extract training instances of workflow behavior. A workflow discovery tool [86] mines activity event logs to create workflow models for intelligent agents. With workflow models as a reference for mission state, planning and scheduling algorithms [76] can recommend security policy actions to information security managers. Security managers interface with intelligent agents using a network tasking order (NTO) [106] to

assign ‘weights of effort’ to particular missions and optimize security-performance tradeoffs. The NTO specifies an action schema to operate a security policy engine adapting policy based on behavior analysis and risk assessment.

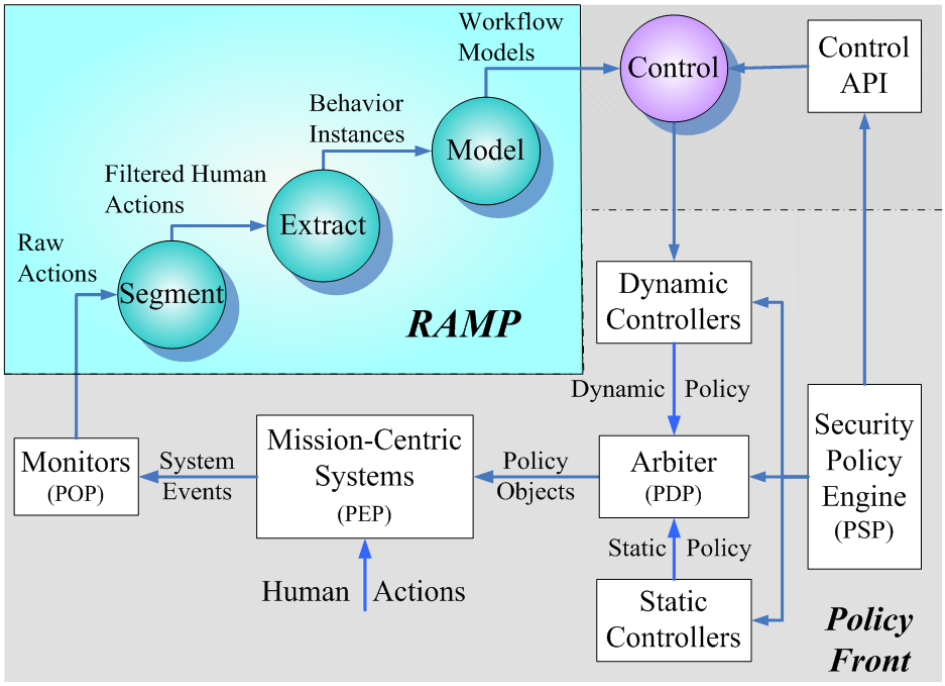

Figure 3.4: Risk Aware, Mission Parameterized (RAMP) policy framework

Workflow behavior auditing prototypes and applications are vital to demonstrate the value of process mining for mission situational awareness. By integrating the algorithms in this dissertation and beyond, prototype systems can entice mission owners to share data from real world environments leading to improved future studies.

7.4. Conclusion

The path towards mission situational awareness is feasible with the ability to audit the organizational expectation of workflow behaviors with the actual and

observable activities in mission centric collaboration. In the battle rhythm, implicit or explicit workflows specify and mandate activities to enable mission success. However, the challenges posed by dynamic mission states, raw and unstructured event logs and inaccurate timestamps make the conventional approach of process mining inadequate. In this dissertation, we addressed these deficiencies by creating three algorithms and an experimental testbed to exhibit the feasibility of process mining as a workflow behavior auditing technology for mission centric collaboration. Workflow shift detection is addressed in the WSDA algorithm using a change point detection scheme of divergent probability distributions between long term and short term observation windows. Unstructured event logs become structured in an automated fashion using string processing techniques for workflow instance extraction. The flower chain discovery algorithm infers the flower chain network for timestamp distorted event logs. Finally, for long term testing and development, we introduced the WASP simulator, the first of its kind in the process mining field.

The future of workflow behavior auditing depends upon continued development of algorithms that can mitigate logging uncertainty. New data sources and system prototypes will also propel next generation mission situational awareness. In doing so, commanders will possess new tools to assess and optimize workflows and missions in the battle space environment.

REFERENCES

[1] J. Choobineh, G. Dhillon, M.R. Grimaila, and J. Rees. "Management of Information Security: Challenges and Research Directions." *Communications of the Association for Information Systems*, vol. 20, no. 57. 2007.

[2] W.M.P. van der Aalst. "Process-Aware Information Systems: Lessons to be Learned from Process Mining." *Transactions on Petri Nets and Other Models of Concurrency II*, Springer, 2009. pp. 1–26.

[3] D.S. Alberts, J.J. Garstka, and F.P. Stein. *Network Centric Warfare: Developing and Leveraging Information Superiority.* Technical Report, Assistant Secretary of Defense, Washington D.C., 2000.

[4] M.R. Grimaila, R.F. Mills, M. Haas, and D. Kelly. "Mission Assurance: Issues and Challenges." in *Proceedings of the 2010 International Conference on Security and Management*, Las Vegas, Nevada, July 12-15, 2010.

[5] W.M.P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer-Verlag, New York, 2011.

[6] R. Sandhu, R. Boppana, R. Krishnan, J. Reich, T. Wolff, and J. Zachry. "Towards a Discipline of Mission-Aware Cloud Computing." in *Proceedings of the 2010 ACM Workshop on Cloud Computing Security*, New York, New York, 2010. pp. 13–18.

[7] M. Blaze, S. Kannan, I. Lee, O. Sokolsky, J.M. Smith, A.D. Keromytis, and W. Lee. "Dynamic Trust Management." *Computer*, vol. 42, no. 2, 2009. pp. 44–52.

[8] M. Schwartz. *Department of Defense Contractors in Iraq and Afghanistan: Background and Analysis*. Diane Publishing, Jul 2010.

[9] L. Sly. "Ceremony Formally Marks End of Coalition Effort in Iraq." *Los Angeles Times*, 2 Jan 2010.

[10] S.M. George., et al. "DistressNet: A Wireless Ad Hoc and Sensor Network Architecture for Situation Management in Disaster Response." *IEEE Communications Magazine*, vol 48, no. 3, 2010. pp. 128-136.

[11] United States, Department of Defense. "Operation Unified Response: Support to Haiti Earthquake Relief 2010." [Online] *Available: http://www.southcom.mil.* [Accessed: 30-Sep-2013].

[12] M.W. Wynne. "Flying and Fighting in Cyberspace." *Air and Space Power Journal*, vol. 21, no. 1, Spring 2007. pp. 6-8.

[13] M. Linderman, B. Siegel, D. Ouellet, J. Brichacek, S. Haines, G. Chase, and J. O'May. *A Reference Model for Information Management to Support Coalition Information Sharing Needs.* Technical Report, Air Force Research Lab, Rome, New York. 2005.

[14] United States, Department of the Air Force. *Service Oriented Information Management.* Technical Report, Air Force Research Lab, Rome, New York, 2013.

[15] S. Kilner, "Case Studies of Process Mining." *IBM Systems Magazine*, Mar 2013.

[16] R. Charette. "U.S. Air Force Blows $1 Billion on Failed ERP Project." *IEEE Spectrum*, 15 Nov 2012.

[17] United States. "Defense Business Modernization." *Fiscal Year 2013 President's Budget Request*, Department of Defense Chief Information Officer, Program and Budget Office, March 2012.

[18] W.M.P. van der Aalst, A. Adriansyah, A.K.A. de Medeiros, F. Arcieri, T. Baier, T. Blickle, J.C. Bose, P. van den Brand, R. Brandtjen, and J. Buijs. "Process Mining Manifesto." *Business Process Management Workshops*, Springer Berlin Heidelberg, 2012. pp. 169–194.

[19] United States. Department of Defense. *Joint Publication 3-33: Joint Force Headquarters.* Washington, DoD, Nov 2010.

[20] K. Conner, P. Lambertson, and M. Roberson. *Analyzing the Air Operations Center (AOC) Air Tasking Order (ATO) Process Using Theory of Constraints (TOC)*. Masters Thesis, Air Force Institute of Technology, March 2005.

[21] J. Gantz and D. Reinsel. "The Digital Universe in 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East." *IDC iView: IDC Analyze the Future*, 2012.

[22] T. Mather, S. Kumaraswamy, and S. Latif. *Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance*. O'Reilly, 2009.

[23] G. Brunette and R. Mogull. "Security Guidance for Critical Areas of Focus in Cloud Computing." *Cloud Security Alliance*, vol. 2, no. 1, 2009. pp. 1–76.

[24] R.P.J.C. Bose, W.M.P. van der Aalst, I. Žliobaitė, and M. Pechenizkiy. "Handling Concept Drift in Process Mining." *Advanced Information Systems Engineering*, H. Mouratidis and C. Rolland, Eds. Springer Berlin-Heidelberg, 2011. pp. 391–405.

[25] C.W. Günther. *Process Mining in Flexible Environments*. Ph.D. Dissertation, Technische Universiteit Eindhoven, 2009.

[26] H. Schonenberg, et al. "Process Flexibility: A Survey of Contemporary Approaches." *Advances in Enterprise Engineering I,* Springer Berlin-Heidelberg, 2008. pp. 16–30.

[27] R.J.C. Bose, W.M. van der Aalst, I. Zliobaite, and M. Pechenizkiy. "Dealing With Concept Drifts in Process Mining: A Case Study in a Dutch Municipality." Technical Report, Technische Universiteit Eindhoven, 2009.

[28] B.F. van Dongen and W.M. van der Aalst. "A Meta Model for Process Mining Data." in *Proceedings of the CAiSE'05 Workshops*, vol. 2, Porto, Portugal, 2005. pp. 309-320.

[29] H.M.W. Verbeek, J.C. Buijs, B.F. van Dongen, and W.M. van der Aalst. "XES, XESame, and proM 6." *Information Systems Evolution*, Springer, 2011. pp. 60–75.

[30] C.W. Günther and W.M. van der Aalst. "A Generic Import Framework for Process Event Logs." *Business Process Management Workshops*, 2006. pp. 81–92.

[31] "Nitro by Fluxicon - Converting Event Logs for Process Mining" [Online]. *Available: http://fluxicon.com/nitro/*. [Accessed: 20-Aug-2013].

[32] C.W. Günther and W.M. van der Aalst. *Mining Activity Clusters from Low-Level Event Logs.* Technical Report, School for Ops Management and Logistics, 2006.

[33] G. Greco, A. Guzzo, L. Ponieri, and D. Sacca. "Discovering Expressive Process Models by Clustering Log Traces." *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 8, 2006. pp. 1010–1027.

[34] R.P.J.C. Bose and W.M. van der Aalst. “Abstractions in Process Mining: A Taxonomy of Patterns.” *Business Process Management*, Springer Berlin-Heidelberg, 2009. pp. 159–175.

[35] E. Barboni, J.F. Ladry, D. Navarre, P. Palanque, and M. Winckler. “Beyond Modeling: an Integrated Environment Supporting Co-execution of Tasks and Systems Models.” in *Proceedings of the 2nd Symposium on Engineering Interactive Computing Systems*, 2010. pp. 165–174.

[36] C. Martinie, P. Palanque, D. Navarre, M. Winckler, and E. Poupart. “Model Based Training: An Approach Supporting Operability of Critical Interactive Systems.” in *Proceedings of the 3rd ACM Symposium on Engineering Interactive Computing Systems*, 2011. pp. 53–62.

[37] A. Hassan, D. Martin, P. Flora, P. Mansfield, and D. Dietz. “An Industrial Case Study of Customizing Operational Profiles using Log Compression.” in *Proceedings of the 30th International Conference on Software Engineering*, 2008. pp. 713–723.

[38] M. Nagappan, K. Wu, and M.A. Vouk. “Efficiently Extracting Operational Profiles from Execution Logs using Suffix Arrays.” in *Proceedings of the 20th International Symposium on Software Reliability Engineering*, 2009. pp. 41–50.

[39] K. Kent and M. Souppaya. *Guide to Computer Security Log Management*. NIST Special Publication, 2006. pp. 800–892.

[40] W.M. van der Aalst, T. Weijters, and L. Maruster. "Workflow Mining: Discovering Process Models from Event Logs." *IEEE Transactions on Knowledge and Data Engineering*, 2004.

[41] A. Rozinat, M. Veloso, and W.M.P. van der Aalst. *Using Hidden Markov Models to Evaluate the Quality of Discovered Process Models.* BPM Center Report, 2008.

[42] G. Khodabandelou, C. Hug, R. Deneckere, and C. Salinesi. "Supervised Intentional Process Models Discovery using Hidden Markov Models." in *Proceedings of the 7th International Conference on Research Challenges in Information Science,* Paris, France, May 2013.

[43] G.A. da Silva and D.R. Ferreira. "Applying Hidden Markov Models to Process Mining." in *Proceedings of the 4th Iberian Conference of Information Systems and Technologies*, 2009.

[44] R. Agrawal, D. Gunopulos, and F. Leymann. *Mining Process Models from Workflow Logs.* Springer, 1998.

[45] W.M. van der Aalst, V. Rubin, B.F. van Dongen, E. Kindler, and C.W. Günther. *Process Mining: A Two-step Approach using Transition Systems and Regions*. Technical Report, Business Process Mining Center, 2006.

[46] A.A. de Medeiros and A. Weijters. "Genetic process mining." *Applications and Theory of Petri Nets*, Springer Verlag, Lecture Notes in Computer Science, vol. 3536, 2005.

[47] B.F. van Dongen, N. Busi, G. Pinna, and W.M. van der Aalst. *An Iterative Algorithm for Applying the Theory of Regions in Process Mining.* Technical Report, Research School for Operations Management and Logistics, 2007.

[48] R. Bergenthum, J. Desel, R. Lorenz, and S. Mauser. "Process Mining based on Regions of Languages." *Business Process Management*, Springer, 2007, pp. 375–383.

[49] A. Weijters, W.M. van der Aalst, and A.A. de Medeiros. *Process Mining with the Heuristics Miner Algorithm*. Technical Report, Technische Universiteit Eindhoven, 2006.

[50] A.K.A. de Medeiros, B.F. van Dongen, W.M. van der Aalst, and A. Weijters. "Process Mining for Ubiquitous Mobile Systems: an Overview and a Concrete Algorithm." *Ubiquitous Mobile Information and Collaboration Systems*, Springer, 2005. pp. 151–165.

[51] B.F. van Dongen, A.K.A. de Medeiros, and L. Wen. "Process Mining: Overview and Outlook of Petri Net Discovery Algorithms." *Transactions on Petri Nets and Other Models of Concurrency II*, Springer Berlin Heidelberg, 2009. pp. 225–242.

[52] W.M. van der Aalst and A.K.A. de Medeiros. "Process Mining and Security: Detecting Anomalous Process Executions and Checking Process Conformance." *Electronic Notes in Theoretical Computer Science*, vol. 121, 2005. pp. 3–21.

[53] W.S. Yang and S.Y. Hwang. "A Process-Mining Framework for the Detection of Healthcare Fraud and Abuse." *Expert Systems with Applications*, vol. 31, no. 1, Jul. 2006. pp. 56–68.

[54] R. Accorsi, C. Wonnemann, and T. Stocker. “Towards Forensic Data Flow Analysis of Business Process Logs.” in *Proceedings of the Sixth International Conference on IT Security Incident Management and IT Forensics*, 2011. pp. 3–20.

[55] R. Accorsi and T. Stocker. “On the Exploitation of Process Mining for Security Audits: The Conformance Checking Case.” in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, 2012. pp. 1709–1716.

[56] R. Accorsi, T. Stocker, and G. Müller. “On the Exploitation of Process Mining for Security Audits: The Process Discovery Case.” in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, 2013. pp. 1462–1468.

[57] A. Baumgrass. “Deriving Current State RBAC Models from Event Logs.” in *Proceedings of the Sixth International Conference on Availability, Reliability and Security*, 2011. pp. 667–672.

[58] A. Baumgrass, T. Baier, J. Mendling, and M. Strembeck. “Conformance Checking of RBAC Policies in Process-Aware Information Systems.” in *Business Process Management Workshops*, 2012. pp. 435–446.

[59] A. Baumgrass, S. Schefer-Wenzl, and M. Strembeck. “Deriving Process-Related RBAC Models from Process Execution Histories.” in *Proceedings of the 36th Annual Computer Software and Applications Conference Workshops*, 2012. pp. 421–426.

[60] S. Schefer-Wenzl, M. Strembeck, J. Mendling, and A. Baumgrass. “Detecting and Resolving Conflicts of Mutual Exclusion and Binding Constraints in a Business

Process Context." *On the Move to Meaningful Internet Systems*, Springer, 2011. pp. 329–346.

[61] L. Lamport. "Time, Clocks, and the Ordering of Events in a Distributed System." *Communications of the ACM*, vol. 21, no. 7, 1978. pp. 558–565.

[62] A. Rogge-Solti, R.S. Mans, W.M. van der Aalst, and M. Weske. *Repairing Event Logs Using Stochastic Process Models.* Technical Report, University of Potsdam, 2013.

[63] "Top 5 Data Quality Problems for Process Mining — Flux Capacitor." [Online]. *Available: http://fluxicon.com/blog/2011/06/data-quality-process-mining/.* [Accessed: 15-Aug-2013].

[64] R.J.C. Bose, R.S. Mans, and W.M. van der Aalst. "Wanna Improve Process Mining Results?" in *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining*. 2013. pp 127-134.

[65] K. Jensen, L.M. Kristensen, and L.Wells. "Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems." *International Journal on Software Tools for Technology Transfer*, vol. 9, no. 3, 2007. pp. 213–254.

[66] F. Bause and P. S. Kritzinger. *Stochastic Petri Nets*. Springer, 2002.

[67] A. Rozinat and W.M. van der Aalst. "Decision Mining in ProM." *Business Process Management*. Springer Berlin-Heidelberg, 2006. pp. 420-425.

[68] J. Pecarina and J.C. Liu. "APSAT: A Framework for Modeling and Analysis of Workflow Dynamics in Mission Centric Systems." in *Proceedings of the 2012*

Conference on Collaboration Technologies and Systems, Westminster, Colorado, 2012, pp. 575–582.

[69] P. Wohed, W.M. van der Aalst, M. Dumas, A.H. ter Hofstede, and N. Russell. On the Suitability of BPMN for Business Process Modelling. *Business Process Management*. Springer Berlin Heidelberg, 2006. pp. 161-176

[70] W.M. van der Aalst and A.H. ter Hofstede. "YAWL: Yet Another Workflow Language." *Information Systems*, vol. 30, no. 4, 2005. pp. 245–275.

[71] G. Combs. "Wireshark: Network Protocol Analyzer." *Wireshark Version 0.99*. 2008.

[72] C.R. Hertel. *Implementing CIFS: The Common Internet File System*. Prentice Hall, 2004.

[73] M. Basseville and I.V. Nikiforov. "Detection of Abrupt Changes: Theory and Applications." *Journal of the Royal Statistical Society-Series A Statistics in Society*, vol. 158, no. 1, 1995.

[74] R. Ihaka and R. Gentleman. "R: A Language for Data Analysis and Graphics." *Journal of Computational and Graphical Statistics*, vol. 5, no. 3, 1996. pp. 299–314.

[75] R.A. Wagner and M.J. Fischer. "The String-to-string Correction Problem." *Journal of the ACM*, vol. 21, no. 1, 1974. pp. 168–173.

[76] S.J. Russell and P. Norvig. *Artificial intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, 1995.

[77] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM press, New York, 1999.

[78] R. Kolpakov, G. Bana, and G. Kucherov. "MREPS: Efficient and Flexible Detection of Tandem Repeats in DNA." *Nucleic Acids Research*, vol. 31, no. 13, 2003. pp. 3672–3678.

[79] E. Ukkonen. "Finding Approximate Patterns in Strings." *Journal of Algorithms*, vol. 6, no. 1, 1985. pp. 132–137.

[80] H. Hyyrö. "A Bit Vector Algorithm for Computing Levenshtein and Damerau Edit Distances." *Nordic Journal of Computing*, vol. 10, no. 1, 2003. pp. 29–39.

[81] J. Pecarina and J.C. Liu. "Behavior Instance Extraction for Risk Aware Control in Mission Centric Systems." in *Proceedings of the 3rd International Conference on Cognitive Methods in Situation Awareness and Decision Support*, San Diego, California, 2013. pp.45-50.

[82] A. Oliner, A. Ganapathi, and W. Xu. "Advances and Challenges in Log Analysis." *Communications of the ACM*, vol. 55, no. 2, 2012. pp. 55–61.

[83] K. Arvind. "Probabilistic Clock Synchronization in Distributed Systems." *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, no. 5, 1994. pp. 474–487.

[84] J.E. Elson and D. Estrin. "Time Synchronization in Wireless Sensor Networks." in *Proceedings of the 15th International Parallel and Distributed Processing Symposium*, University of California, Los Angeles, 2001.

[85] A. Rozinat, M.T. Wynn, W.M. van der Aalst, A.H. ter Hofstede, and C.J. Fidge. "Workflow Simulation for Support." *Data and Knowledge Engineering*, vol. 68, no. 9, 2009. pp. 834–850.

[86] W.M. van der Aalst, B.F. van Dongen, C.W. Günther, R.S. Mans, A.A. De Medeiros, A. Rozinat, V. Rubin, M. Song, H.M.W. Verbeek, and A. Weijters. "ProM 4.0: Comprehensive Support for Real Process Analysis." *Petri Nets and Other Models of Concurrency*, Springer, 2007, pp. 484–494.

[87] M. Westergaard and L. M. Kristensen. "The Access/CPN Framework: A Tool for Interacting with the CPN Tools Simulator." *Applications and Theory of Petri Nets*, Springer, 2009. pp. 313–322.

[88] A.A. De Medeiros and C.W. Günther. "Process Mining: Using CPN Tools to Create Test Logs for Mining Algorithms." in *Proceedings of the 6th Workshop on the Practical Use of Coloured Petri Nets and CPN tools*, vol. 576, 2005.

[89] A. Rozinat, A.A. de Medeiros, C.W. Günther, A. Weijters, and W.M. van der Aalst. *Towards an evaluation framework for process mining algorithms*. Technical Report, Research School for Operations Management and Logistics, 2007.

[90] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. "The Role of Trust Management in Distributed Systems Security." *Secure Internet Programming*, 1999. pp. 185–210.

[91] M. Blaze, J. Feigenbaum, and J. Lacy. "Decentralized Trust Management." in *Proceedings of the IEEE Symposium on Security and Privacy*, 1996. pp. 164–173.

[92] M. Blaze, J. Feigenbaum, and A.D. Keromytis. “KeyNote: Trust Management for Public Key Infrastructures.” *Security Protocols,* Springer Berlin Heidelberg, 1999.

[93] C.J. McDonald, J.M. Overhage, M. Barnes, G. Schadow, L. Blevins, P.R. Dexter, and B. Mamlin. “The Indiana Network for Patient Care.” *Health Affairs,* vol. 24, no. 5, 2005.

[94] J. Kohl and C. Neuman. “RFC 1510: The Kerberos Network Authentication Service (V5).” *Request For Comments,* vol. 11. 1993.

[95] R. Sandhu, et al. “The NIST Model for Role Based Access Control: Towards a Unified Standard.” in *Proceedings of the 5th Symposium on Access Control Models and Technology*, vol. 26, 2000. pp. 47–63.

[96] J. Pecarina, S. Pu, and J.C. Liu. “SAPPHIRE: Anonymity for Enhanced Control and Private Collaboration in Healthcare Clouds.” in *Proceedings of the 4th International Conference on Cloud Computing Technology and Science*, Taipei, Taiwan, 2012. pp. 99–106.

[97] W.H. Winsborough, K.E. Seamons, and V.E. Jones. "Automated Trust Negotiation.” in *Proceedings of the DARPA Information Survivability Conference and Exposition,* vol. 1. IEEE, 2000. pp. 88–102.

[98] E. Bertino, E. Ferrari, and A. Squicciarini. “Trust Negotiations: Concepts, Systems, and Languages.” *Computing in Science & Engineering*, vol. 6, no. 4, 2004. pp. 27–34.

[99] A. Sahai and B. Waters. “Fuzzy Identity Based Encryption.” *Advances in Cryptology*, 2005. pp. 557-565.

[100] V. Goyal, O. Pandey, A. Sahai, and B. Waters. “Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data.” in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, 2006. pp. 89–98.

[101] L. Wang, D. Wijesekera, and S. Jajodia. “A Logic-Based Framework for Attribute Based Access Control.” in *Proceedings of the 2004 ACM Workshop on Formal Methods in Security Engineering*, 2004. pp. 45–55.

[102] J. Park and R. Sandhu. “The UCON ABC Usage Control Model.” *ACM Transactions on Information and System Security*, vol. 7, no. 1, 2004. pp. 128–174.

[103] K. Knorr. “Dynamic Access Control through Petri Net Workflows.” in *Proceedings of the 16th Annual Conference on Computer Security Applications,* 2000. pp. 159–167.

[104] Nagios. [Online]. *Available: http://www.nagios.org/*. [Accessed: 20-Aug-2013].

[105] Splunk. [Online]. *Available: http://www.splunk.com/*. [Accessed: 20-Aug-2013].

[106] J.M. Pecarina. *Creating an Agent Based Framework to Maximize Information Utility.* Masters Thesis, Air Force Institute of Technology, March 2008.