An integrative process mining approach to mine discrete event simulation model from event data

Yan Wang

université de BORDEAUX

THÈSE PRÉSENTÉE

POUR OBTENIR LE GRADE DE

DOCTEUR DE

L'UNIVERSITÉ DE BORDEAUX

ÉCOLE DOCTORALE DES SCIENCES PHYSIQUES ET DE L'INGÉNIEUR

SPECIALITÉ: PRODUCTIQUE

Par Yan WANG

AN INTEGRATIVE PROCESS MINING APPROACH TO MINE DISCRETE EVENT SIMULATION MODEL FROM EVENT DATA

Sous la direction de : David CHEN
(Co-directeur : Grégory ZACHAREWICZ, Mamadou Kaba TRAORÉ)

Soutenue le 12 Octobre 2018

Membres du jury :

1.	M. ARCHIMÈDE, Bernard	Professeur, École nationale d'ingénieurs de Tarbes	Président
2.	M. SANTUCCI, Jean François	Professeur, Université de Corse Pascal Paoli	Rapporteur
3.	Mme. DIAMANTINI, Claudia	Professeur, Università Politecnica delle Marche	Rapporteur
4.	M. CHEN, David	Professeur, Université de Bordeaux	Directeur
5.	M. ZACHAREWICZ, Grégory	Professeur, École des Mines d'Alès	Co-directeur
6.	M. TRAORÉ, Mamadou Kaba	MCF (H.D.R), Université Clermont Auvergne	Co-directeur

This thesis is dedicated to my wife Mouna and my parents.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisors Prof. Gregory Zacharewicz, Prof. Mamadou Kaba Traore and Prof. David Chen for their continuous support of my PhD study and related research, for their patience, motivation and immense knowledge. Their guidance helped me to make the research and write the thesis. I could not have imagined having a better supervisor and mentor for my PhD study.

Besides my supervisors, I would like to thank Prof. Jean-Paul Bourrieres, Dr. Youssef Bouanan and Dr. Alioune Fall for their insightful comments. They have helped me to extend the knowledge of my PhD study.

My sincere thanks also goes to China Scholarship Council for supporting me a four years funding of my PhD study.

At last, I would like to thank to my wife in France for her suggestions of writing my PhD thesis and her encouragement. I also thank to my parents in China for their support in my PhD study. Without the love of my wife and my parents, I will not be able to finish this thesis.

Résumé

L'inférence d'un système, par la reconstruction de la structure à partir de l'analyse de son comportement, est reconnue comme un problème critique. Dans la théorie des systèmes, la structure et le comportement se situent aux extrémités de la hiérarchie qui définit la connaissance du système. L'inférence d'un système peut être également considérée comme l'escalade de la hiérarchie depuis la connaissance de bas niveau vers la connaissance de plus haut niveau. Ceci n'est possible que sous des conditions maitrisées et justifiées. Dans cette thèse, une nouvelle méthode d'inférence de système est proposée. La méthode proposée étend la technique Process Mining pour extraire des connaissances depuis les données des événements du système. Les aspects de modularité, de fréquence et de synchronisation peuvent être extraits des données. Ils sont intégrés ensemble pour construire un modèle Fuzzy-Discrete Event System Specification (Fuzzy-DEVS). La méthode proposée, également appelée méthode D2FD (Data to Fuzzy-DEVS), comprend trois étapes: (1) l'extraction depuis des journaux d'évènements (registres) obtenus à partir des données générées par le système en utilisant une approche conceptuelle; (2) la découverte d'un système de transition, en utilisant des techniques de découverte de processus; (3) l'intégration de méthodes Fuzzy pour générer automatiquement un modèle Fuzzy-DEVS à partir du système de transition. La dernière étape est de l'implémenter cette contribution en tant que plugin dans l'environnement Process Mining Framework (ProM). Afin de valider les modèles construits, une approximation de modèle basée sur le morphisme et une méthode prédictive intégrée à Granger Causality sont proposées. Deux études de cas sont présentées dans lesquelles le modèle Fuzzy-DEVS est déduit à partir de données réelles, où l'outil SimStudio est utilisé pour sa simulation. Les modèles ainsi construits et les résultats de simulation sont validés par comparaison à d'autres modèles.

Mots-clés: Inférence de système, Process Mining (Découverte de Processus), Fuzzy-DEVS (DEVS Flous), Modélisation et simulation, Validation de modèle.

Abstract

System inference, i.e., the building of system structure from system behavior, is widely recognized as a critical challenging issue. In System Theory, structure and behavior are at the extreme sides of the hierarchy that defines knowledge about the system. System inference is known as climbing the hierarchy from less to more knowledge. In addition, it is possible only under justifying conditions. In this thesis, a new system inference method is proposed. The proposed method extends the process mining technique to extract knowledge from event data and to represent complex systems. The modularity, frequency and timing aspects can be extracted from the data. They are integrated together to construct the Fuzzy Discrete Event System Specification (Fuzzy-DEVS) model. The proposed method is also called D2FD (Data to Fuzzy-DEVS) method, and consists of three stages: (1) extraction of event logs from event data by using the conceptual structure; (2) discovery of a transition system, using process discovery techniques; (3) integration of fuzzy methods to automatically generate a Fuzzy-DEVS model from the transition system. The last stage is implemented as a plugin in the Process Mining Framework (ProM) environment. In order to validate constructed models, morphism-based model approximation and predictive method integrated with Granger Causality are proposed. Two case studies are presented in which Fuzzy-DEVS model is inferred from real life data, and the SimStudio tool is used for its simulation. The constructed models and simulation results are validated by comparing to other models.

Keywords: System Inference, Process Mining, Fuzzy-DEVS, Modeling and Simulation, Model Validation

Résumé

Cette thèse s'inscrit dans le cadre de la modélisation et de la simulation des processus d'entreprise et vise à contribuer à l'ingénierie des processus d'entreprise et à ses utilisations dans les entreprises. La modélisation d'entreprise traite du processus de compréhension d'une entreprise et de l'amélioration de ses performances grâce à la création de modèles d'entreprise. Parmi les langages de modélisation d'entreprise, le langage de modélisation de processus est l'un des plus utilisés pour construire des modèles de processus. Dans la majorité des cas, les modèles de processus sont construits sur la base d'hypothèses d'experts ou de la participation à des entretiens. Les techniques de Process Mining (PM) utilisées dans cette thèse fournissent une méthode ascendante pour construire des modèles. PM consiste à découvrir, surveiller et améliorer des processus réels en extrayant des connaissances à partir des journaux d'événements. Si des modèles de processus existent dans une entreprise, PM permet de vérifier si le modèle établi est correct et permet également de compléter ou d'améliorer les processus existants. Si les modèles de processus n'existent pas dans une entreprise, PM fournit un moyen complémentaire de construire des modèles de processus.

L'objectif principal de cette thèse est d'exploiter un modèle de simulation à partir des données d'événement. Avec le développement de l'entreprise, les systèmes d'information deviennent de plus en plus grands. Les données enregistrées dans les systèmes d'information augmentent. L'événement décrit le déroulement d'une activité et il s'agit d'une partie des données. L'un des grands défis aujourd'hui consiste à extraire des informations utiles de ces données d'événement. Pour cette raison, le modèle de processus est construit à partir d'événements en appliquant des techniques d'exploration de processus. La simulation du modèle de processus fournit des résultats liés au processus pour la découverte de systèmes réels et de recherches expérimentales. La validation détermine que les résultats de

la simulation ont une précision suffisante pour l'entreprise. Cette thèse se concentre sur la découverte du processus. La découverte de processus en tant que partie intégrante de l'exploration de processus fournit des techniques pour construire des modèles de processus à partir de journaux d'événements. Le modèle de simulation est basé sur Discrete Event System Specification (DEVS). Il fournit un cadre général pour décrire les systèmes complexes. Dans la théorie des systèmes, la structure et le comportement se situent aux extrêmes de la hiérarchie qui définit la connaissance du système. De plus, DEVS offre une définition formelle du simulateur.

La principale problématique de recherche dans cette thèse est l'inférence du système. L'inférence de système est définie par le fait que le système existe et nous essayons de générer la structure à partir d'observations de son comportement. En outre, il est difficile de collecter des données, car PM n'accepte que les journaux d'événements appropriés. Ceci correspond à la corrélation des événements, à l'instance de processus et à l'étendue des données. Il existe de nombreux langages de modélisation de processus utilisés dans PM. Le réseau de Petri est l'un des plus utilisés. Comparé à DEVS, le temps est considéré comme une extension ou une perspective dans PM. En outre, la modularité est insuffisante, ce qui rend difficile la conception de modèles hiérarchiques. Comparé à d'autres méthodes de découverte de processus, nous pouvons découvrir que l'alpha-algorithme est capable de fournir une simultanéité mais ne prend pas en compte la fréquence. Le modèle n'est pas solide. Le modèle construit par l'exploitation minière régionale est bien ajusté et précis. En intégrant le concept de région, le modèle est généralisé. Cependant, le modèle ressemble au spaghetti et est difficile à comprendre. La simplification est un problème. L'exploration heuristique prend en compte la fréquence. En ajustant les seuils, le modèle peut tre simplifié. Cette méthode peut se concentrer sur ce qui est important et ce qui n'est pas important. D'autres techniques, telles que l'extraction floue, suppriment les activités de simplification. Mineur inductif peu fréquent (IiM) définit le seuil pour contrôler le filtrage. Chaque méthode de découverte de processus est indépendante. Nous devons simplement en trouver un qui explique le mieux les données d'événement observées. Enfin, le modèle découvert nécessite la validation opérationnelle. La validation opérationnelle est définie comme la

détermination que le comportement de sortie du modèle a une précision suffisante pour la fonction à laquelle il est destiné.

Dans cette thèse, une approche intégrée est proposée pour découvrir un modèle Fuzzy Discrete Event System Specification (Fuzzy-DEVS) à partir de données d'événement, également appelé méthode D2FD (Data to Fuzzy-DEVS). La méthode D2FD comprend trois étapes: (1) extraction des journaux d'événements à partir de données d'événement en utilisant structure conceptuelle; (2) découverte d'un système de transition utilisant un processus techniques de découverte; (3) intégration de méthodes floues pour automatiquement générer un modèle Fuzzy-DEVS à partir du système de transition. Dans la première étape, les données d'événement sont définies et nous proposons une méthode en cinq étapes. Cette méthode en cinq étapes comprend définition des objectifs, identification des relations, identification des valeurs, sélection des instances de processus et cartographie générale. Dans la deuxième étape, nous réutilisons une partie de l'extraction régionale pour découvrir le système de transition à partir des journaux d'événements. Dans la troisième étape, une approche basée sur la région améliorée propose de découvrir la région correspondent - aux états de ing Fuzzy-devs. Dépendance La méthode est utilisée pour produire la transition interne floue, transition externe floue et fonction de sortie floue; Le contrôleur de temps Fuzzy adapté (AFTC) est utilisé pour obtenir une fonction d'avance de temps flou. Pour exécuter le modèle Fuzzy-DEVS, les mesures de possibilité et la sortie finale de AFTC sont appliqués. Le résultat final de AFTC est déduit de la moyenne pondérée méthode à partir de méthodes de défuzzification. Le cluster flou est appliqué aux fonctions de Modèle couplé Fuzzy-DEVS. La méthode D2FD est implémentée en tant que plug-in dans l'environnement Process Mining Framework (ProM). L'outil de simulation SimStudio est intégré pour sa simulation. Afin de valider les modèles construits, une approximation du modèle basé sur le morphisme et une méthode de prévision intégrée à la causalité de Granger sont proposées. Deux études de cas sont présentées dans lesquelles le modèle Fuzzy-DEVS est déduit de données réelles. Une de l'agence néerlandaise d'assurance des employés et une autre du groupe Rabobank ICT (Technologies de l'information et de la communication). Les modèles construits et les résultats de simulation sont validés par la comparaison à d'autres modèles.

Mots-clés: Inférence de système, Process Mining (Découverte de Processus), Fuzzy-DEVS (DEVS Flous), Modélisation et simulation, Validation de modèle.

Table of Content

List of Figures

List of Tables

Prologue

This research has been carried out in the research team "Enterprise Modeling and Engineering" at the IMS laboratory of University of Bordeaux. The thesis presented in this document fits within the frame of the modeling and simulation of business processes of enterprises and aims at contributing to enterprise process engineering and its uses in enterprises.

Since 1970's, many enterprise modeling techniques have been developed. Among them, the process modeling language is one of the most used in enterprises to elaborate process models. In most of the cases, process models are built top-down by modeling experts with the help of a methodology through a series of interviews to collect information. The process mining technique proposed in this thesis is a complementary bottom-up approach to process modeling methodologies. In an enterprise where process models already exist, applying process mining allows verifying if established process models are correctly followed as defined in the models. It also allows complementing or improving existing processes with additional discovered process data. In an enterprise where there are no established process models, process mining technique can be used in a complementary way together with an enterprise modeling methodology to collect process information in order to build process models.

General Introduction

With the development of enterprises, information systems are becoming bigger and bigger. For example, people and organizations depend more and more on the information and devices on the Internet. Information systems are also becoming more and more complex as these systems can consist of multiple and heterogeneous components and intricate interactions among these components. This leads to a big explosion of multitudes of data recorded in the information systems. Nevertheless, most of the data are unstructured and organizations have problems to analyze these data. One of the big challenges today is to extract valuable information from the data in the information systems. Data mining (Hand, 2007) provides approaches to make data understandable and useful to the data users. However, multitude of events are also recorded in the information systems. Data mining has difficulties to deal with process-related information. Process mining (Van der Aalst, 2011)(Van der Aalst, 2016), as a relatively young area, provides techniques to extract useful information from events and improve the business processes. Compared with data mining, process mining not only has control flow discovery, but also has conformance checking which animates the business process. Process mining connects classical data analytic like data mining with Business Process Management(BPM) (Weske, 2012).

BPM is the discipline that combines knowledge from information technology and knowledge from management sciences and applies this to operational business processes. In BPM, most of business processes are recorded by events. Organizations are more and more eager to manage, control and support their business process. One of the big challenges for organization is improving business process to reduce time and save resources. By modeling a business process and simulating it, organizations can make better decisions on this improvement. The model in simulation has a fundamental role, as it makes the real system to be studied more accessible and comprehensible. The simulation can be considered as a valid method of inquiring, or rather a procedure for discovering real system and experimental research (Piu, 2010).

The original process of developing models and simulations has three steps. A conceptual model is designed by a subject matter expert from careful consideration of a problem and its domain. Then, it is realized via a source code simulation through the implementation of interfaces, data structures and algorithms. Finally, the output of the simulation for a set of test cases is validated against historical data or other trusted sources. As a result, modeling and simulation are based on a set of fundamental assumptions (Pace, 2000). In the process mining, we do not build models based on expert assumptions, but often build process models from events. The mined process model is connected to real system and does not provide an idealized view on the processes. In order to evaluate the process model whether it is a good reflection of the real process, we have to deal with four forces. Four forces include fitness, simplicity, precision and generalization (Van der Aalst, 2011). Fitness refers to the ability to explain observed behavior. The model should be as simple as possible. Precision means the model should not allow all kinds of behavior unrelated to the event data that we have seen. At the same time, the model should not be over-fitting. Process mining needs to find a balance between these four forces.

After modeling and simulation, the developer or the user of the model, the organizations often concern about the correctness of this model. This concern is addressed through model verification and model validation. Model verification is related to the computer program and the implementation. Model validation considers about the distance between the accuracy and the intended purpose of the model.

Chapter 1

Research Problem

1.1 Introduction

The main objective of the research is to design a method to mine a simulation model from event data, so that the organization can extract useful process information from data. During the research, system inference is the main research problem. Three more problems are found and need to be solved. The collection of data is a big challenge. It is also necessary to select an appropriate formalism for simulation model to represent business process. At last, the mined simulation model needs methods for validation.

1.2 Problem Statement

System Theory (Simon, 1991) provides a fundamental framework to understand dynamical systems. In such a framework, a system is characterized by its structure and its behavior. All the knowledge about the system can be organized in a 4-level hierarchy (Klir, 2013). This hierarchy is depicted in Figure 1.1. It is organized as follows:

- the source level identifies a portion of the real world we are going to observe and measure;
- the data level corresponds to the set of measurements made on the system from its observation;
- the generative level uses formulas or equations to constitute a knowledge;
- the structure level describes the component systems that are interconnected together to form the entire system.

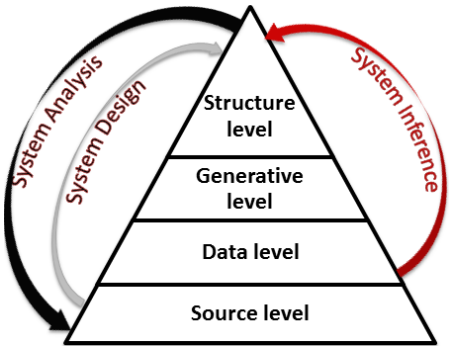


Figure 1.1: Levels of system knowledge and system problems.

The system structure is defined by the top levels of the hierarchy and the system behavior by the bottom levels. Actually, if we go from the high level to the low level, we will get less knowledge. Conversely, if we go from the low level to the high level, we will get more knowledge. Models as the abstract representations use a simplified and straightforward way to describe the real world. The model in the structure level can provide more useful information to the organization.

Moving between the levels of system knowledge in Figure 1.1, there are three different kinds of system problems. They are organized as follows:

- System analysis, we know the system structure (existing or hypothetical) and we try to generate its data;
- System design, the system does not exist yet and we are investigating the alternative structures for a completely new system;
- System inference, the system exists and we are trying to generate its structure from known evidence of its behavior. This has been called climbing the hill by Zeigler (Zeigler et al., 2000). Note that a slight but very significant difference between system design and system inference is the system existence or not, prior to the study.

For system analysis, most of researchers are working on developing models. Most of the models are based on expert assumptions. But the models they build by hand are disconnected from real system. Sometimes these models are not precise. For system design, there are more than one way to design a model. It is not easy to design a model which satisfies the organizations. System inference starts from the data level to the structure level. Some criteria are justified in the data level. These conditions make unique solutions of generating models. System inference is identified

as big challenge in the modeling and simulation. It is recognized as the main problem in this thesis.

Process mining (Van der Aalst, 2011)(Van der Aalst, 2016) provides techniques to infer from knowledge given at data level, corresponding knowledge at the structure level. The idea of process mining is to discover, monitor and improve real processes by extracting knowledge from event logs readily available in real systems. In the process mining, process discovery takes the event logs to produce a process model. While making a process model, the model user is trying to view a process. Moreover, the discovered process model can be used for animation and specification. Process discovery can provide potential solutions for system inference, but there are three problems which need to be handled.

1.2.1 Challenge of Collecting Data

In the process mining, only proper event logs are accepted to construct process models. There is a big gap between event logs and event data. The challenge is not the syntactical conversion but to extract event logs from a variety of data sources for example internet of events, a database system, a comma-separated values (CSV) file, a transaction log, an enterprise resource planning (ERP) system, a message log. First, the challenge requires event correlation. The underlying relationship needs to be explored so that events are related to each other. Second, the process instance needs to be identified. In a document-based business process, the process instance to be selected is related to at least one document. Third, the scoping of event data is a problem. Enterprise or organizations can make thousands of tables which contain a lot of business information. It is more efficient to locate on the required and interesting data.

1.2.2 Extension of Model Mining in Process Mining

In the process mining, Petri Net (Petri and Reisig, 2008) is often identified as the resulting process models. Apart from Petri Net, there are also other different kinds of process models. However, most of these process models do not consider about time. Time is identified as one of extensions in the process mining. In addition, there is a lack of modularity, making the design of hierarchical models difficult to realize. Therefore, process mining shows limitations in inferring complex systems.

1.2.3 Validation of Mined Model

Fuzzy-DEVS model needs to be validated by the people and the organizations. However, the discovered Fuzzy-DEVS model can get only one simulation result, and this result is not always validated based on different enterprise requirements. There is a technological barrier which makes people or enterprises unable to get information through computer. The reason may be the use of different method and techniques to represent information. It is necessary to propose some new integrated approaches for the validation of Fuzzy-DEVS models.

1.3 Thesis Contribution

In this thesis, we propose an integrative method from event data to Fuzzy-DEVS model (D2FD). DEVS is selected as the simulation model. Moreover, we choose Fuzzy-DEVS to express the imprecision from event data. The general structure of D2FD method is shown in Figure 1.2. This method deals with system inference. The event data comes from the data level and Fuzzy-DEVS model locates on the structure level in Figure 1.1. D2FD method considers about frequency and time, and deals with modularity problem. On the one hand, the frequency and the time information are considered and extracted from event data. On the another hand, the modularity is observed by conceptual structure of event data. The validation approaches make the model compatible for the people and the organization. The mined Fuzzy-DEVS model and its simulation results can help to improve business process and make better decisions.

1.4 Thesis Outline

Chapter 2 gives state of the art including DEVS framework, process mining and fuzzy logic. DEVS graphical notations and Fuzzy-DEVS formalism are explained in the DEVS framework. Event logs, process models and Two Phase Approach are explained in the process mining. Fuzzy cluster is explained in the fuzzy logic. The following chapters are presented from top to bottom in Figure 1.2. From event data to event log, chapter 3 introduces a five-steps method represented in the first red block. The definitions of event data are discussed at first. This five-steps method includes setting up goals, identification of relationships, identification of values, selection of process instance and general mapping. The first part of Two Phase Approach in process mining is generally presented at the end of chapter 3 represented in the

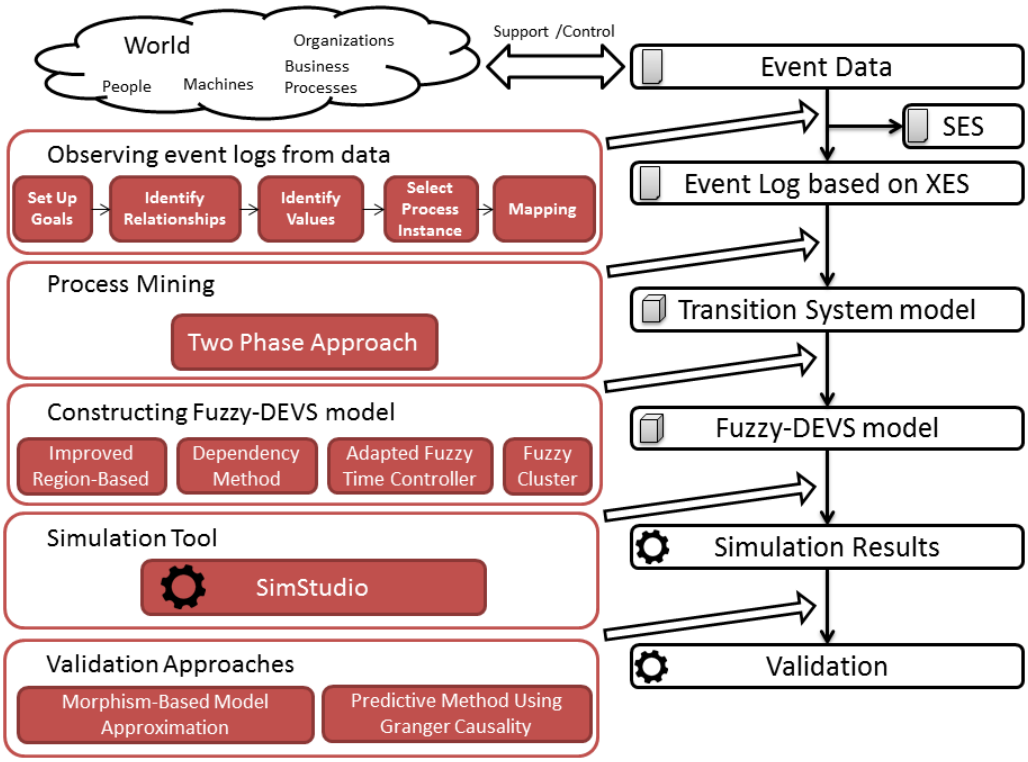


Figure 1.2: General structure of D2FD method.

second red block. Chapter 4 explains the main part of D2FD method represented in the third red block. An improved region-based approach, dependency method and AFTC are proposed for the discovery of Fuzzy-DEVS atomic model. Fuzzy cluster is proposed for the construction of Fuzzy-DEVS coupled model. The implementation of D2FD method and two case studies are presented in chapter 5. The process mining tool ProM is the implementation environment. The simulation tool SimStudio is integrated for its simulation represented in the fourth red block. In chapter 6, two validation approaches, morphism-based model approximation and predictive method using Granger causality are proposed. The case study used in chapter 5 evaluates these two methods. The results of this case study are validated by comparing with other models. At last, chapter 6 concludes this thesis. The limitation of D2FD method and perspectives are discussed. The publications are listed.

Chapter 2

State of the Art

2.1 Introduction

The main objective of this thesis is to discover a Fuzzy-DEVS model from event data. This research is involved in the fields of DEVS framework, process mining and fuzzy logic. Section 2.2 gives the background of framework of modeling and simulation, system formalisms and system specification to describe dynamic systems. Ontology extends the knowledge base of modeling and simulation. The conceptual structure inherited from System Entity Structure (SES) is used for D2FD method. In section 2.3, event logs are the starting point. Process mining provides different methods to discover different process models from event logs. In section 2.4, the basic concept of fuzzy logic is fuzzy sets. Fuzzy sets as a controller can be applied in the Fuzzy-DEVS. Fuzzy control as one of the main methods combines fuzzification methods and defuzzification methods. Fuzzy cluster as one part of cluster analysis is introduced at the end of section 2.4.

2.2 DEVS Framework

The theory of modeling and simulation has been proposed by Bernard P. Zeigler in the 1970s (Zeigler et al., 2000). This theory provides a general framework with several theoretical foundations. There are two main parts of these foundations:

- System specification formalisms: define the types of system models which are either continuous or discrete. These formalisms are defined by a set of concepts and principles.
- Levels of system specification: provide the levels which describe the behaviors and the mechanism of system. These levels correspond to the levels of system

knowledge in Figure 1.1.

DEVS is one of the system specification formalisms. DEVS model can be identified as moore model which its output depends on the state (Wagner, 2005). We first give the framework of modeling and simulation with some essential concepts. Then we present DEVS and some related studies. According to different extensions of DEVS, we focus on Fuzzy-DEVS. The levels of system specification which corresponds to DEVS are explained later. Ontology of modeling and simulation is discussed in the end.

2.2.1 Framework of Modeling and Simulation

Bernard P. Zeigler (Zeigler et al., 2000) proposed a framework for modeling and simulation as shown in Figure 2.1. This framework provides some basic concepts which consists of entities and their relationships. It makes everyone understand more about modeling and simulation. The entities include source system, behavior databases, experimental frame, model and simulator.

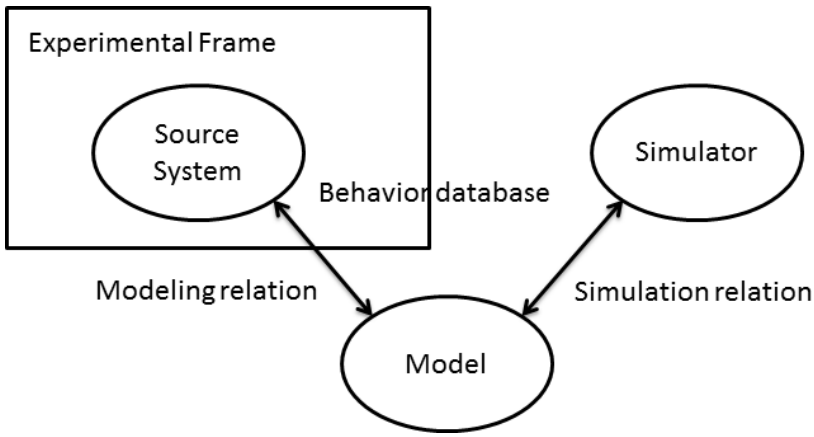


Figure 2.1: Framework for modeling and simulation (Zeigler et al., 2000).

In Table 2.1, source system and behavior database are related to the source level and data level in the levels of system knowledge respectively. In particular, behavior database is collected from source system in real system and specified under the experimental frame. The experimental frame not only defines the type of data elements that will go into database, but also extracts interesting data by interacting with interesting system. For example, the frame with interesting data can be classified by generator which generates input segments to system, acceptor which monitors the experiments and makes evaluations, and transducer which make analysis on output segments of

Table 2.1: Definition of entities and corresponding system specification hierarchy in Table 2.2

Basic entity	Definition	Corresponding system specification hierarchy
Source system	Data in real system or human made	Level 0
Behavior database	Data collected from source system	Level 1
Experimental frame	The specification of observed and experimental system	Levels 3 and 4
Model	Instructions for generating and representing data	Levels 3 and 4
Simulator	The computational tool to generate the behavior of the model	Levels 4

the system (Zeigler and Sarjoughian, 2003). Inside the experimental frame, experimental base defines a set of all possible experiments which can be carried out from the real system, experimental condition defines a set of conditions corresponding to one specific experiment, and experimental control defines experiment parameters which only affect the execution of model computations in a simulation (Elzas, 1984).

There are many ways to define a model. A model can be designed from science, technology and mathematics. In mathematical logic, a model defines a set which has a collection of axioms and the operations and relations between them (Bender, 2012). In terms of system specification, a model specifies a real system. Model as the representation of a real phenomenon provides a clear semantic that everyone can understand. Base model is the closest to real system which accounts for all the input/output behavior of the real system (Elzas, 1984). In Figure 2.1, the modeling relation defines which part of system is related to model. Correspondingly, the model can be distinguished between behavioral model and structural model. The behavioral model can be described by a triple (Willems, 1989):

$$\Sigma = (T, W, B) \tag{2.1}$$

Where T is the set of time over which the system evolves. W is the signal space in which the variables whose evolution in time take on their value. $B \subseteq W^T$ is behavior which is the set of signals that are compatible with the laws which govern the system (W^T represents the set of all signals.). The main object in the behavioral model is the behavior which is the set of all signals compatible with the system and there is no difference between input and output variables.

The behavior model as the basic component can be coupled together to form a high-level specification model - structural model. An important concept of structure is decomposition which means that a system can be split into component systems at a lower level. Conversely, the concept of composition defines the components systems which are coupled together to form the entire system. These concepts are based on "closed under composition" (Zeigler et al., 1999), which gives the mathematical proof of the equivalence of behavioral and structural models. For every coupled model, an equivalent atomic model (component) can be constructed.

A simulation model consists of a set of instructions, rules, equations or constraints. A simulator is a system calculator which follows the relevance of instructions to execute a model and produce behaviors. It probably can be a computer, a network or abstractly a logic processor, an algorithm. The simulator focuses on the simulation of structural models. The simulation relation is related to simulation correctness which guaranties that a simulator correctly simulates the model. The validity of models, systems and experimental frame defines the degree on which a model faithfully represent its system.

2.2.2 System Specification Formalisms

A system is characterized by its structure and its behavior. The external behavior consists of the input time segment and the output time segment. The internal structure consists of state and state transition as well as the mapping between state and output. According to hierarchical system composition, the component systems are coupled together to form the larger ones. The coupling provides a binding from output port to input port. There are three advantages of coupling feature:

1. It can be developed and tested as a stand-alone unit.

2. It can be stored in a model repository and reactivated when needed.

3. It can be reused in any applications when it has appropriate behavior.

Models can be represented by a variety of formalisms. Based on mathematical concepts and properties, there are two dimensions of formal model which are space and time (Oren, 1987). The space defines that the variables can take finite or infinite values. It can be distinguished between continuous and discrete. The continuous variable of model takes the real values. The discrete variable of model has finite set of values. Time can also be distinguished between continuous and discrete. The continuous time of model describes the time with real values. The discrete time of

model defines the time is temporal and the set of values is finite. By combining the classification of space and time, we can get four kinds of formalisms as shown in Figure 2.2 (Zacharewicz, 2006).

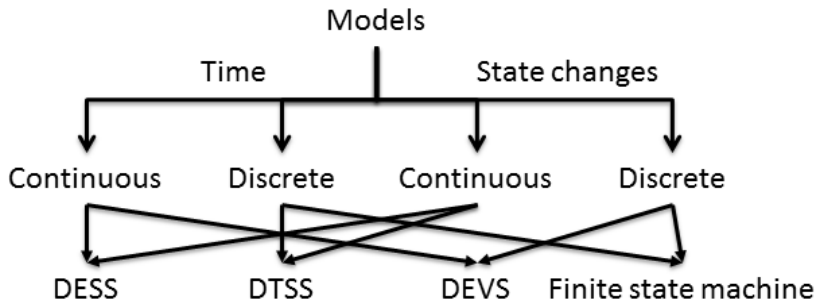


Figure 2.2: Different system specification formalisms.

- DESS (Differential Equation System Specification): has continuous time and continuous states. The differential equation defines the change of the state.
- DTSS (Discrete Time System Specification): has discrete time and continuous states. The change of the state is based on difference equation.
- DEVS (Discrete Event System Specification): has discrete states and continuous time. The change of the state depends on the event and each event can occur at any time.
- Finite state machine: has discrete states and time. It is also called synchronous finite state machine.

2.2.2.1 Discrete Event System Specification (DEVS)

The DEVS formalism for modeling and simulation is based on discrete events, and provides a framework with mathematical concepts based on the set theory and the system theory concepts to describe the structure and the behavior of the system (Zeigler et al., 2000). It provides a universal way to represent dynamic systems regardless of the application area.

The two main formalisms are the Classical DEVS (CDEVS) and the Parallel DEVS (PDEVS). Most of extensions are based on one of these two. A real system modeled by CDEVS is described as a number of connected behavioral (atomic) and structural (coupled) components (Zeigler et al., 2000). An atomic model D in CDEVS is defined as follows:

$$D =< X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta > \tag{2.2}$$

Where X is the set of input values; S is the set of states; Y is the set of output values; δ_{int} is the internal transition function; δ_{ext} is the external transition function; λ is the output function; ta is the duration function. Each atomic model has the duration specified by $ta(s)$ where s is the current state. When the elapsed time $e = ta(s)$, the state duration expires and the atomic model will send the output $\lambda(s)$ and performs an internal transition to a new state specified by $\delta_{int}(s)$. However, state transition can also happen due to arrival of an external event which will place the model into a new state specified by $\delta_{ext}(s, e, x)$ where x is the input value. The time advance function $ta(s)$ can take any real value from 0 to ∞. A state with $ta(s)$ value of 0 is called transient state, and on the other hand, if $ta(s)$ is equal to ∞ the state is said to be passive, in which the system will remain in this state until receiving an external event.

A coupled model N in CDEVS is defined as follows:

$$N =< X, Y, D, EIC, EOC, IC, select > \tag{2.3}$$

Where both X and Y respectively define the sets of input and output events. D is a set of indices for the components. The external input coupling EIC specifies the connections between external and component inputs, while the external output coupling EOC describes the connections between component and external outputs. The connections between the components themselves are defined by the internal coupling IC. The tie-breaking function *select* is used to solve collisions between multiple components. When two or more components execute their internal transition at the same time, this function makes one selection from them. The coupling and transformation between separating atomic models make it possible to construct a more complicated hierarchical model.

PDEVS was presented in (Chow and Zeigler, 1994) to better support for parallellism. Compared with CDEVS atomic model, a new function confluent transition function δ_{conf} is added in PDEVS. This function is used to solve collisions between δ_{int} and δ_{ext}. The collisions happen when an external event comes and an internal transition is triggered at the same time. Compared with CDEVS coupled model, the select function is deleted in PDEVS. The models are executed in parallel instead of sequential.

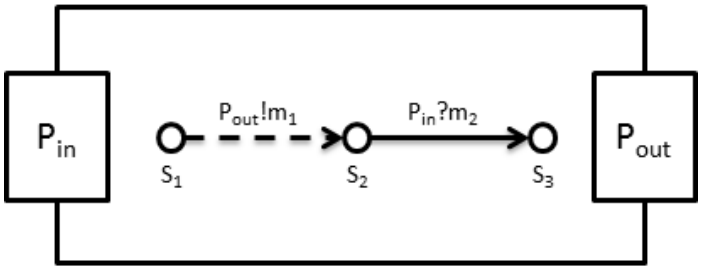


Figure 2.3: An example of DEVS graphical notation.

The formal properties of DEVS formalism not only has closure under composition, but also has universality and uniqueness (Zeigler and Sarjoughian, 2013). Universality means that DEVS is able to generate all kinds of discrete event behaviors. Uniqueness means that DEVS can represent their canonical internal structures isomorphically.

2.2.2.2 Graphical Notation of DEVS

Song et al. (Song and Kim, 1994) propose a graphical notation to represent the DEVS atomic model. The DEVS atomic model is represented by a box with input and output ports. These ports are the entrances and exits of messages. The messages which come from the entrance are the input events on X and the messages from the exit are the output events on Y. An input event $p?m$ means that a message m is coming from the input port p. Similarly, an output event $p!m$ means that a message m is going to the output port p. This output event $p!m$ also represents the output function λ(s). Each state variable in the state set S of DEVS is inside the graphical box and each one has an unique name. The behavior of the DEVS atomic model is represented by an activity or a state transition diagram. This diagram consists of nodes and two kinds of arc. The nodes represent the activity or the state. The dotted arc denotes the internal transition and the solid arc denotes the external transition.

Figure 2.3 shows an example of an DEVS atomic model. The model is represented by a box with two ports P_{in} and P_{out}. There are three states S_1, S_2 and S_3 represented by nodes. The dotted arc represents an internal transition $\delta_{int}(S_1) = S_2$ and a corresponding output function $\lambda(S_1) = P_{out}!m_1$. After the time of the state S_1 is finished, the output event $P_{out}!m_1$ goes to the port P_{out} and the state changes to S_2. The solid arc represents an external transition $\delta_{ext}(S_2, P_{in}?m_2) = S_3$. The time of the state of S_2 is infinite. When the input event $P_{in}?m_2$ comes from the port P_{in}, the external transition δ_{ext} is executed and the state changes to S_3.

2.2.2.3 Extensions of DEVS and Related Studies

The extensions of DEVS are established based on different conditions when applying DEVS formalism. DEVS can be extended to Dynamic Structure DEVS and Generalized-DEVS. In spite of Fuzzy-DEVS, DEVS can be extended to Cell DEVS, Real Time DEVS, Min-Max DEVS, Symbolic DEVS, Stochastic DEVS and Fuzzy-DEVS.

Dynamic Structure DEVS (DSDEVS)(Barros, 1995) proposes to use a network executive in each coupled model to solve dynamical problems. This network executive can collect all the messages in the system and then initiate a restructuring. However, without network executive, models cannot change themselves. DSDEVS can be distinguished between DynDEVS (Uhrmacher, 2001) and DSDE (Barros, 1997). In DynDEVS, every model can change itself and its structure even if it is not mentioned to network executive. DSDE integrates PDEVS with DSDEVS and it is implemented in VLE (explained in section 5.2.4).

Generalized-DEVS (G-DEVS) defines abstractions of signals with piecewise polynomial trajectories (Giambiasi et al., 2001). G-DEVS together with HLA (High Level Architecture) is able to present a distributed Workflow Reference Model (Zacharewicz et al., 2008).

Cell DEVS (Wainer and Giambiasi, 2001) integrates DEVS with Cellular Automata. Cellular Automata divides the model into cells as well as the time. A time delay is required for Cell DEVS. The downside is that the time becomes discrete which increases the granularity of the simulation. Cell DEVS is mainly interesting in the natural science and social network (Bouanan et al., 2014).

Real Time DEVS (Cho and Kim, 1998) proposes a real-time, interactive simulation method for discrete event models. This method is working on an object-oriented environment. The relation of activity in which the activity is scheduled for execution can be used for real time simulation (Hu, 2004).

Min-Max DEVS (Hamri et al., 2006) proposes to use interval to represent a possible value for the actual delay in DEVS which is also called min-max delay. This method is used for internal transition and external transition in DEVS. Fuzzy-DEVS for the transitions between states takes uncertainty into account and Min-Max DEVS for the lifetime of the state considers about imprecision (Bisgambiglia, 2008).

Symbolic DEVS (Lee and Chi, 2005) proposes to observe the time information from the symbolic for example traffic signal control variables and traffic information. The simulation is based on real-time.

Stochastic DEVS (Castro et al., 2008) proposes to apply probability on the internal transition and external transition of DEVS. Since Fuzzy-DEVS is related to possibility, the comparison between possibility and probability is shown in section 2.4.2.

2.2.2.4 Fuzzy-DEVS Formalism

The formalism of Fuzzy-DEVS (Kwon et al., 1996) applies fuzzy set theory on the original DEVS formalism which is explained in section 2.2.2.1. Since four functions of the original DEVS formalism are defined on crisp sets, Fuzzy-DEVS generalizes the four functions on fuzzy sets. The DEVS model can be distinguished by DEVS atomic model and DEVS coupled model. The four functions are all in the DEVS atomic model for example, internal transition, external transition, output function and time advance function. A fuzzy atomic model $\widetilde{D}$ is specified as follow:

$$\widetilde{D} =< X, Y, S, \widetilde{\delta}_{int}, \widetilde{\delta}_{ext}, \widetilde{\lambda}, \widetilde{ta} > \tag{2.4}$$

Where

- X: the set of input values.
- Y: the set of output values.
- S: the set of states.
- $\widetilde{\delta}_{int}$: $S \times S \rightarrow [0, 1]$, fuzzy internal transition function.
- $\widetilde{\delta}_{ext}$: $Q \times X \times S \times S \rightarrow [0, 1]$, fuzzy external transition function, Q = {(s, e) | s $\in$ S, 0 $\leq$ e $\leq$ ta(s)} where ta(s) is the defuzzified value of $\widetilde{ta}$.
- $\widetilde{\lambda}$: $S \times Y \rightarrow [0, 1]$, fuzzy output function.
- $\widetilde{ta}$: $S \times \widetilde{A} \rightarrow [0, 1]$, fuzzy time advance function. $\widetilde{A}$ is the set of fuzzy linguistic numbers.

Here, the DEVS concepts of internal transition, external transition, output function and time advance function, are integrated with fuzzy set logic. Fuzzy set logic is introduced in section 2.4.1. Fuzzy internal transition and fuzzy external transition provide state transitions with fuzzy relation which represents the possibility of each state transition. $\widetilde{\delta}_{int}(s_t, s_{t+1})$ has a membership function on $S \times S$ which is between 0 and 1. When the elapsed time reaches to the defuuzified value $ta(s_t)$, this membership

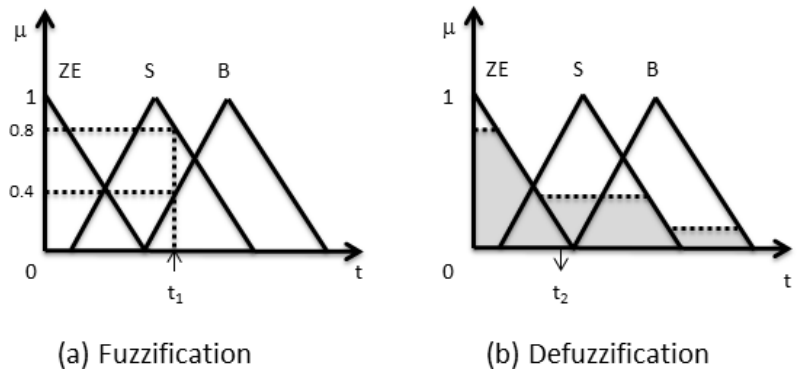


Figure 2.4: An example of fuzzy time advance.

function gives the possibility of the transition from the present state s_t to the next state s_{t+1}. Respectively, $\widetilde{\delta}_{int}(s_t, x_t, s_{t+1})$ has a membership function on $Q \times X \times S \times S$ which is between 0 and 1. When the input event x_t comes and the elapsed time e is smaller than ta(s_t) (in another words, the input event applies before the time advance of the state s_t), this membership function gives the possibility of the transition from the present state s_t to the next state s_{t+1}. Fuzzy output function also provides the output function with fuzzy relation to represent possibilities of output event. According to the fuzzy internal transition, fuzzy output function has a membership function on $S \times Y$ which is between 0 and 1. When the elapsed time reaches to $ta(s_t)$, $\widetilde{\lambda}(s_t, y_t)$ has a possibility of the output event y_t from the state s_t. $\widetilde{\lambda}(s_t, y_t)$ is the grade that the output event y_t goes to the corresponding port in the Fuzzy-DEVS atomic model.

Fuzzy time advance function is extended to a fuzzy set $\widetilde{A}$ which represents fuzzy linguistic numbers. These fuzzy linguistic numbers may be such as "small" and "big". The fuzzy time advance functions has a membership function on $S \times \widetilde{A}$ which is between 0 and 1. $\widetilde{ta}(s, \widetilde{a})$ represents the possibility of the tima advance value of the state s associated with the linguistic numbers $\widetilde{a}$. For example, if $\widetilde{a}$ = "small", $\widetilde{ta}(s, \widetilde{a}) = 0.3$. If $\widetilde{a}$ = "big", $\widetilde{ta}(s, \widetilde{a}) = 0.7$. As a result, the state can have more than one time advance value with an appropriate possibility. The time advance function of the state can not be decided precisely which is also called uncertainty. When the actual time advance value is needed for simulation , the defuzzification methods can be used to transform the fuzzy time advance values into crisp values. The methods of defuzzification are discussed in section 2.4.4. These crisp values can be fuzzed again to be submitted to customers. Figure 2.4 shows an example of fuzzy time advance. The fuzzy numbers "ZE", "S", "B" represent the linguistic terms "near zero", "small", "big". The time advance value t_1 in Figure 2.4(a) has a fuzzy time value of "small" with the possibility of 0.8 and "big" with the possibility of 0.4. In Figure 2.4(b), if the

current state has a fuzzy time value of "near zero" with the possibility of 0.8, "small" with the possibility of 0.4 and "big" with the possibility of 0.2, we can get a crisp time value t_2 using defuzzification method. The defuzzification method is centroid method which calculates all the gray part of Figure 2.4(b).

The second form of Fuzzy-DEVS model is Fuzzy-DEVS coupled model. The coupled model connects all the atomic model together to form the entire system model. The formalism of Fuzzy-DEVS coupled model is the same as DEVS coupled model in section 2.2.2.1.

2.2.3 Levels of System Specification

Table 2.2: System specification hierarchy

Level	Specification name	Corresponding level of knowledge	What we know at this level
0	Observation frame	Source system	How to stimulate the system with inputs; what variable to measure and how to observe them over a time base.
1	I/O behavior	Data system	Time-indexed data collected from a source system; consists of input/output pairs.
2	I/O function		Given an initial state, every input stimulus produces a unique output.
3	State transition	Generative system	Given a state and an input, what is the state after the input stimulus is over; what output event is generated by a state.
4	Coupled component	Structure system	Components are coupled together. The components can be specified at lower levels or can even be structure systems themselves.

The 4-level hierarchy in Figure 1.1 proposed by Klir describes the levels of system knowledge. Meanwhile, B.P. Zeigler (Zeigler et al., 2000) proposed the hierarchy of system specification as shown in Table 2.2. Source system is observed from real world in level 0. Then the inputs and outputs are extracted in level 1. The initial state is necessary in level 2 so the system is not only decided by inputs and outputs but also

by initial state. This level aims to generate a conceptual model implemented on a computer that it is able to return to the initial state (Zacharewicz, 2006). Then state transition mechanisms are described at level 3 and coupling feature at level 4. The main difference between these two hierarchies is the simulation context in which the hierarchy of system specification can describe how the system behave over time. The hierarchy of specification in Table 2.2 corresponds to DEVS framework. For example, level 1 specifies the inputs and outputs of DEVS. Level 2 and 3 specify the states of DEVS.

2.2.4 Ontology for Modeling and Simulation

Modeling and simulation, regarding its philosophy computational and conceptual aspects, is used as a tool to solve technical or managerial problems. There are three important concepts for modeling and simulation: ontology, epistemology and teleology (Tolk, 2012). Ontology is a branch of metaphysics handling the nature of being. Methodological ontologies capture the knowledge of paradigms and methods. Referential ontologies can ensure to use the same terminology and relations when addressing problems. In the modeling and simulation, the methodological ontology refers to "how to model" and the referential ontology refers to "what to model" (Michiardi and Molva, 2002). The ontology of intelligent modeling and simulation applications is important because the two generated models from experts can be misaligned on the conceptual level. This misalignment can be caused by difference in resolution scope and structure or cultural and organizational biases. Epistemology provides the constraints on both methodological and referential ontology of modeling and simulation. It is able to identify the parts which are not appropriate due to the limitations of accessible tools and methods. Teleology is related to the validation that models not only represent knowledge but also gain knowledge. The beautiful ontology is anticipated in which the examples of ontologies are considered by their authors in different arguments as beautiful (D'Aquin and Gangemi, 2011).

DEVS has the modularity which is able to construct hierarchical system models. Applied to ontology engineering, modularity is central not only to reducing the complexity of understanding ontologies but also to maintaining, querying and reasoning over modules (Kutz and Hois, 2012).

The System Entity Structure (SES) (Zeigler and Hammonds, 2007) approach defines an ontological framework to represent modeling and simulation knowledge in a hierarchical manner. Figure 2.5 shows the basic elements of the SES. Entities represent things that exist in the real world. They can be assigned with variables, which

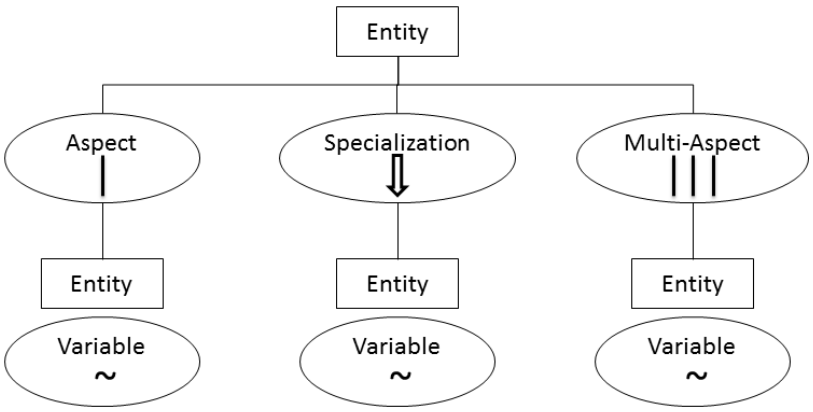


Figure 2.5: Basic representation of SES.

provide values within given ranges and types. Aspects represent ways of decomposing entities into more detailed parts. Multi-aspects are aspects for which the components are all of one kind. Specializations represent categories in specific forms that an entity can assume.

2.3 Process Mining

BPM (Weske, 2012) is widely used as a structured approach toward the goal of improving agility and operational performance. To better understand BPM, we consider Business Process Lifecycle which consists of design and analysis, configuration, enactment and evaluation. One of the phases is the evaluation which uses information available to evaluate and improve business process models and their implementations. However, most of organizations and researchers analyze and build models based on expert assumption rather than quantified collected fact. Most of information is not observed from real system to support evaluation phase and trigger the lifecycle.

Process mining (Van der Aalst, 2011) provides the methods to discover, monitor and improve actual processes by extracting knowledge from event logs readily available in real systems. There are three techniques in process mining as shown in Figure 2.6:

- Discovery: is used to extract an event log with using recorded historical information to produce a model.
- Conformance: is used to check if the event logs from reality conform to the process model.

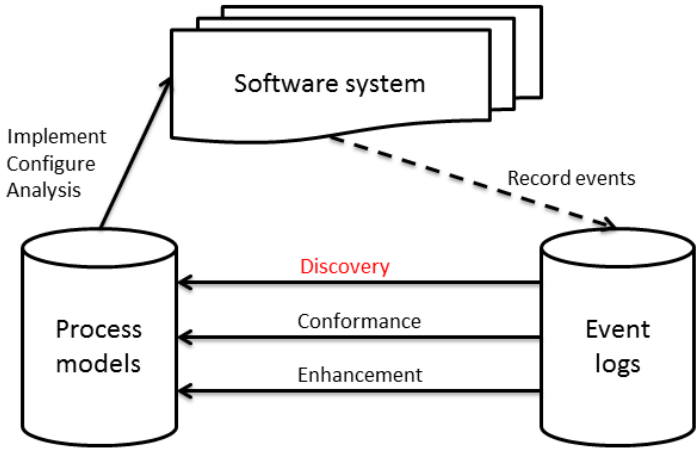


Figure 2.6: General structure of process mining.

- Enhancement: uses information of event logs about the actual process to extend or improve an existing process model.

According to the objective of the thesis, we focus on the process discovery. First, we give an introduction of event logs in section 2.3.1. In section 2.3.2, we present some of process models which are used in the process mining. Four main process discovery methods are introduced in section 2.3.3.

2.3.1 Event Logs

The event logs are based on the XES standard (Günther and Verbeek, 2009). Figure 2.7 shows a complete meta-model of the XES standard using UML (Unified Modeling Language) class diagram. An XES document (i.e., XML file) contains one log which is related to one specific process. A log can contain any number of traces. Each trace describes a sequential list of events corresponding to a particular case. Events present the atomic granules of activity observed from operational process. The log, its traces, and its events may have any number of attributes. Attributes are defined by the type of data value they represent for example string, date, int, float, boolean, ID, list and container. Attributes can be distinguished by nested attributes and global attributes. Nested attributes can provide maximum flexibility. For example attributes can have child attributes when using lists or containers. Global attributes are the attributes which are available and properly defined for all the elements on their respective level all over the document. Besides, the event classifier which is composed of attributes is used as an identity to compare between events. It is mandatory in the XES standard and needs to be defined at first. An extension

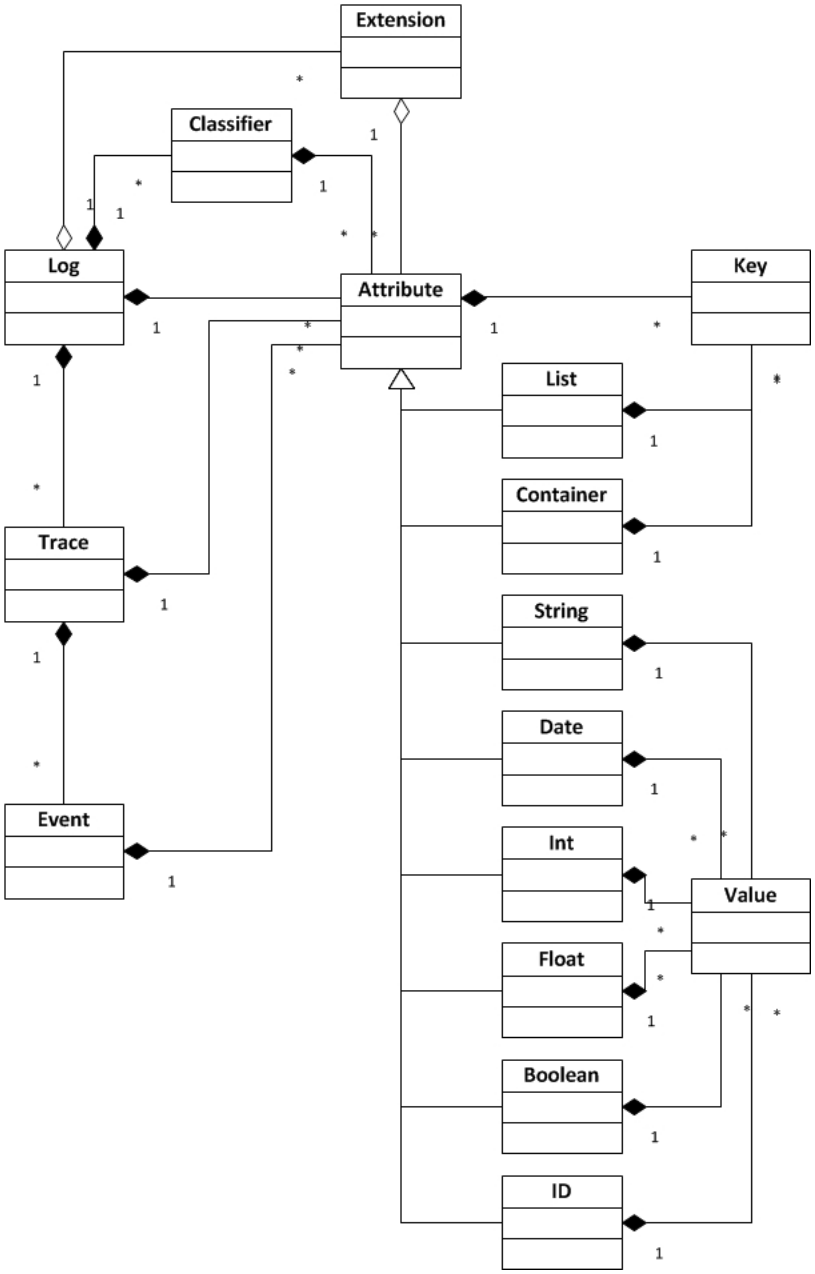


Figure 2.7: Complete meta-model for the XES standard (Günther and Verbeek, 2009).

defines a set of attributes for a specific perspective. The extensions aggregate the log. The concept extension defines an attribute which stores commonly understood names of type elements. Similarly, the time extension stores the time information. The concept extension and time extension are mostly used in this thesis. In addition, there are lifecycle extension (i.e. a lifecycle transition in a model), organizational extension (i.e. events based on human or organizations), semantic extension (events or other elements of the XES type may refer to different concepts), ID extension (i.e.

unique identifiers of elements) and cost extension (i.e. cost information associated with activity). Section 3.3.5 shows an example of XES file.

2.3.2 Process Models

Petri nets (Petri and Reisig, 2008) is mostly used as the resulting process model in the process mining. Petri Nets is one of several mathematical modeling languages for the description of distributed systems. A Petri net is a bipartite graph consisting of place and transition. Arc is used to connect between place and transition. The network structure is static and token flows through the network. Despite the four parameters, the state of a Petri net is determined by the distribution of tokens and is referred as marking. Enabling and firing are the main operating rules. One of the recent studies is using process mining techniques to analyze the career of students and construct Petri Nets models (Cameranesi et al., 2017).

Comparing with DEVS, Petri Nets can be embedded into DEVS because a DEVS model can represent any discrete event behavior (Kočí and Janoušek, 2009). Jacques and Wainer (Jacques and Wainer, 2002) propose an approach of mapping of the Petri Net modeling formalism into the DEVS modeling formalism. This approach is implemented based on the simulation tool CD++ (explained in section 5.2.4). Bazoun et al. (Bazoun et al., 2014) define a transformation approach of BPMN models into DEVS simulation models based on the meta-model approach and describe the enrichment of obtained DEVS models through performance indicators. This approach includes an exhaustive mapping, the transformation architecture and an implementation in the modeling and simulation tool SLMToolBox.

Transition System (TS) is identified as the most basic process modeling notation. The concept of TS (Keller, 1976) is discussed in 1976. A formal definition of a TS is a pair (S, $\rightarrow$) where S is a set of states and $\rightarrow$is a binary relation on S, called the set of transitions. A named TS is a triple (S, $\rightarrow$, Σ) where (S, $\rightarrow$) is a TS and each TS is assigned one or more names in the set Σ.

A visualized TS (Van der Aalst, 2011) is a triplet $TS = (S, A, T)$ where S is the set of states, A is the set of activities, and $T \subseteq S \times A \times S$ is the set of transitions. The states are represented by black circles. There are one initial state S^{start} and one final state S^{end}. In this thesis, we only consider about Finite-State Machine (FSM) where the state space is finite. Each state has a unique label. This label is merely an identifier and has no meaning. Transitions are represented by arcs. Each transition connects two states and is labeled with the name of an activity. Multiple arcs can bear the same label.

Given a TS, one can reason about its behavior. The transition starts in one of the initial states. Any path in the graph starting in such a state corresponds to a possible execution sequence. A path terminates successfully if it ends in one of the final states. A path deadlocks if it reaches a non-final state without any outgoing transition. The TS may live-lock if some transitions are still enabled but it is impossible to reach one of the final states. An example of TS is shown in section 3.4.

WorkFlow nets (WF-nets) (Van der Aalst, 1998) is developed as a subclass of Petri nets. The difference between them is that there are a dedicated source place as input and a dedicated sink place as output. WF-nets is able to describe the start and the end of an business process instance so it is appropriate for business process modeling. Moreover, if we enrich WF-nets with notations, we can get a YAWL (Yet Another Workflow Language) (Ter Hofstede et al., 2009). The purpose of YAWL language is to have a simple language using various descriptive patterns.

A more advanced language is Business Process Modeling Notation (BPMN)[1] which is widely used in the business process modeling. The latest version is 2.0. There are start event and end event. The atomic activity is represented by task. The split and join routes between tasks can be represented by gateways. There are different types of gateways: AND, XOR, OR. The pool and lane can provide more information about organization and actors.

Event-driven Process Chains (EPCs) (Scheer, 1998) is another business process modeling which uses classical notations to describe models. The EPCs is one of the first language which allows for OR split and joins in the notations and the notations of EPCs are the subsets of BPMN and YAWL.

Causal nets is proposed to provide causal dependencies in the process mining (Van der Aalst et al., 2011). The causal dependencies are defined as a relationship between two activities where one is triggering or enabling the other one. In the graphical representation, the nodes represent activities and the arcs represent causal dependencies. The arcs can also be annotated with numbers which represent frequencies. Each node can have a set of possible input bindings and a set of possible output bindings. Obligation is defined like token in Petri nets which starts with start activity and ends with end activity. Input binding removes obligation and output binding generates obligation. Valid binding sequences are conducted by the obligation. A Causal net is a tuple $C = (A, a_i, a_o, D, I, O)$ where:

- $A \subseteq \mathcal{A}$ is a finite set of activities.

[1] https://www.omg.org/spec/BPMN/2.0/About-BPMN/

- $a_i \in A$ is the start activity.
- $a_o \in A$ is the end activity.
- $D \subseteq A \times A$ is the dependency relation.
- $\mathcal{P}(A) = \{A' | A' \subseteq A\}$ is the power-set of A.
- $AS = \{X \subseteq \mathcal{P}(A) | X = \{\varnothing\} \vee \varnothing \notin X\}$ is sets of sets of activities.
- $I \in A \rightarrow AS$ defines the set of possible input bindings each activity.
- $O \in A \rightarrow AS$ defines the set of possible output bindings each activity.

Causal nets is recognized as the output of the heuristic mining in section 2.3.3.3. It fits well with the above process modeling languages. It is able to model XOR, AND and OR without adding more modeling elements.

Process instance model (Diamantini et al., 2016), in the form of instance graph, can be applied in the perspective of organization to generate highly variable process instances. This model focuses on the analysis of individual process instances. Causal relations need to be defined and inferred from the log in order to build instance models.

2.3.3 Process Discovery Methods

In this section, we introduce four of the most relevant process discovery techniques. Every technique in the process mining has its advantage but noise and incompleteness problems always exist. Each method has an independent representation which best explains the observed event data. α-algorithm is one of the first algorithm which is able to discover concurrency. Two Phase Approach is selected to be used in the D2FD method. The first part of transforming from event logs to TS is discussed in section 3.4. And another part of transforming from TS to Petri Nets is discussed in section 2.3.3.2. Heuristic mining has two stages which first learns a dependency graph and then extends to a Causal nets. Genetic process mining is integrated with genetic algorithm which uses evolution to optimize models. Fuzzy mining adaptively simplifies the discovered process models (Günther and Van der Aalst, 2007). The infrequent Inductive Miner (iIM) sets a noise threshold to control filtering in order to get a simpler model. (Cameranesi et al., 2017).

2.3.3.1 α Algorithm

α algorithm (Van der Aalst et al., 2004) provides a basic process discovery approach. The idea of α algorithm is to discover models from event logs which can have loops and parallel parts while guaranteeing certain properties. The inputs of this algorithm focus on the control flow and they ignore the actual timestamps at which the events happen, resources and other data elements. As a result, an event log is converted to a multiset of traces and each trace is a sequence of activity names. The same trace may appear several times. For example, we have an event log L which contains six traces. $L = [(a, b, c, d)^3, (a, c, b, d)^2, (a, e, d)]$. The sequence (a, b, c, d) was executed three times and the sequence (a, c, b, d) was executed two times. The starting point of α algorithm are ordering relations. The relations are defined as follows:

- Direct relation $x > y$: there was at least one case where x is directly followed by y.
- Causality $x \rightarrow y$: x is followed by y but y is never followed by x.
- Parallel $x||y$: x is sometimes followed by y and y is sometimes followed x.
- Choice $x\#y$: x is never directly followed by y and y is never directly followed by x.

These relations are used to discover patterns in the process. The patterns are explained as follows:

- sequence pattern: $a \rightarrow b$
- XOR-split pattern: $a \rightarrow b$, $a \rightarrow c$, and $b\#c$
- XOR-join pattern: $b \rightarrow d$, $c \rightarrow d$ and $b\#c$
- AND-split pattern: $a \rightarrow b$, $a \rightarrow c$, and $b||c$
- AND-join pattern: $b \rightarrow d$, $c \rightarrow d$ and $b||c$

In more detail, it is able to construct a matrix of footprint. Every cell of the matrix has one of these four relationships, i.e. causality in one direction $\rightarrow$, causality in another direction $\leftarrow$, parallel $||$ and choice $\#$. According to the event log L_1, Table 2.3 shows an example of footprint.

Table 2.3: An example of matrix of footprint

	a	b	c	d	e
a	$\#_L$	$\rightarrow_L$	$\rightarrow_L$	$\#_L$	$\rightarrow_L$
b	$\leftarrow_L$	$\#_L$	$\|\|_L$	$\rightarrow_L$	$\#_L$
c	$\leftarrow_L$	$\|\|_L$	$\#_L$	$\rightarrow_L$	$\#_L$
d	$\#_L$	$\leftarrow_L$	$\leftarrow_L$	$\#_L$	$\leftarrow_L$
e	$\leftarrow_L$	$\#_L$	$\#_L$	$\rightarrow_L$	$\#_L$

Once we get the footprint, α algorithm follows eight lines to construct a Petri net (Van der Aalst et al., 2004). Let L be an event log over T. $\alpha(L)$ is defined as follows:

1. $T_L = \{t \in T | \exists_{\sigma \in L} t \in \sigma\}$, each activity corresponds to a transition.
2. $T_I = \{t \in T | \exists_{\sigma \in L} t = first(\sigma)\}$, first elements of each trace corresponds to start activity.
3. $T_O = \{t \in T | \exists_{\sigma \in L} t = last(\sigma)\}$, elements that appear last in a trace corresponds to end activity.
4. $X_L = \{(A, B) | A \subseteq T_L \wedge A \neq \varnothing \wedge B \subseteq T_L \wedge B \neq \varnothing \wedge \forall_{a \in A} \forall_{b \in B} a \rightarrow_L b \wedge \forall_{a1,a2 \in A} a_1 \#_L a_2 \wedge \forall_{b1,b2 \in B} b_1 \#_L b_2\}$, calculate pairs (A, B).
5. $Y_L = \{(A, B) \in X_L | \forall_{(A',B') \in X_L} A \subseteq A' \wedge B \subseteq B' \Rightarrow (A, B) = (A', B')\}$, delete non-maximal pairs (A, B).
6. $P_L = \{p_{(A,B)} | (A, B) \in Y_L\} \cup \{i_L, O_L\}$, determine places $p_{(A,B)}$ from pairs (A, B).
7. $F_L = \{(a, p_{(A,B)}) | (A, B) \in Y_L \wedge a \in A\} \cup \{(p_{(A,B)}, b) | (A, B) \in Y_L \wedge b \in B\} \cup \{(i_L, t) | t \in T_I\} \cup \{(t, o_L) | t \in T_O\}$, determine the flow relation.
8. $\alpha(L) = (P_L, T_L, F_L)$, places P_L, transitions T_L, and arcs F_L in Petri net.

As a basic approach for process discovery, α algorithm has many limitations which is difficult to get a best model. It can have implicit places and loops of length but can be solved through pre-processing. The discovered model is not sound and there are deadlocks inside it. It is not able to represent dependencies and discover transitions with duplicate or invisible labels.

2.3.3.2 Region-Based Mining

In general, region-based mining consists of three parts: Learning TS; Process discovery using state-based regions; Process discovery using language-based regions. Two Phase Approach refers to the first two parts. Since the first part of Two Phase Approach is reused in the D2FD method in section 3.4, we focus on using state-based regions to synthesize a Petri net from it. First we give the definition of state-based region. Let $TS = (S^T, A, T)$ be a TS and $R \subseteq S^T$ be a subset of states. R is a region if for each activity $a \in A$ one of the following conditions hold:

- All transitions $(s_1^T, a, s_2^T) \in T$ enter R, i.e. $s_1^T \notin R$ and $s_2^T \in R$.
- All transitions $(s_1^T, a, s_2^T) \in T$ exit R, i.e. $s_1^T \in R$ and $s_2^T \notin R$.
- All transitions $(s_1^T, a, s_2^T) \in T$ do not cross R, i.e. $s_1^T, s_2^T \in R$ or $s_1^T, s_2^T \notin R$.

Since the region can consist of two small regions, we only have interests on minimal regions. The basic idea is that each minimal region R corresponds to a place p_R in a Petri net. The activities entering the region become Petri-net transitions having p_R as output place, activities leaving the region become output transitions of p_R , and activities that do not cross the region correspond to Petri-net transitions not connected to p_R. As a result, the minimal regions fully encode a Petri net.

2.3.3.3 Heuristic Mining

Heuristic mining algorithm (Weijters and Ribeiro, 2011) takes frequencies and sequences into account and constructs Causal nets. In the first phase, we need to get the direct succession from the event logs and we only consider about causality which is important. The direct succession $a \rightarrow b$ is calculated by the number of times that a was followed by b somewhere in the log. Dependency Method is identified as a more sophisticated measure to discover causality. Thresholds can be set for the minimal number of dependency. Dependency Graph can be generated while including only arcs that meet both thresholds. Frequency and Dependency Method are explained more in detail in section 4.2.2.

In the second phase, we extend a dependency graph with the split join semantics and frequencies to get a Causal net. To discover the split and join, there are two main types of approaches. The first one takes a time window before and after each activity. Based on counting sets of input and output activities, we can determine the bindings. The second one which is more expansive chooses a particular variant of activity that

has a finite number of things. And then we see whether the traces can be replayed properly, assuming input and output bindings. At last, we use certain goal function to choose the best bindings. The frequency indicates how often the corresponding activity appeared in the bindings, connections and activity. The difference between direct succession and frequency is that the bindings can have frequency which can be bigger than direct succession.

2.3.3.4 Genetic Mining

Genetic mining (De Medeiros et al., 2007) provides a non-deterministic approach which is different from the other process mining techniques. From an event log, we can randomly create process models identified as the first generations. Then we can measure the quality. If the quality is already good enough, we can stop and get the final model. Otherwise, we look at the quality to choose the best candidates go to the next round. The so-called crossover operator can be used to recombine multiple candidates and create new models. Mutation can be used to change something in the model. All these methods are random and we can get a new generation of model. Maybe after hundreds or thousands of iterations, we end up with a model with good quality. This algorithm can be very slow if we are dealing with a big amount of data in a real system. But it is extremely flexible to add quality measures.

2.4 Fuzzy Logic

In the fuzzy logic, the basic concept is based on fuzzy sets. Both Possibility Measures and Probability Measures are explained. One of the important fuzzy logic is fuzzy control. Fuzzy control logic can be used to construct an fuzzy inference framework which is composed of fuzzification, defuzzification and fuzzy rules. Defuzzification consists of different mathematical approaches. The method of fuzzy control is discussed with the studies of discrete event modeling and simulation. At last, fuzzy cluster is presented.

2.4.1 Fuzzy Sets

The concept of fuzzy sets, defined by Zadeh (Zadeh, 1996), can be used to change the crisp set of the characteristic function. It provides a logical point of view to deal with the uncertainty. A fuzzy set $\widetilde{F}$ is equivalent to giving a reference set Ω and a

mapping $\mu_F : \Omega \to [0, 1]$. For $\omega \in \Omega$, $\mu_F(\omega)$ is interpreted as the degree of membership of ω in the fuzzy set $\widetilde{F}$. So the fuzzy set can be defined as:

$$\widetilde{F} = \{(\omega, \mu_F(\omega))|\omega \in \Omega, 0 \leqslant \mu_F(\omega) \leqslant 1\} \tag{2.5}$$

This fuzzy set can also be extended to the binary or n-ary. A binary fuzzy set $\widetilde{R}$ is equivalent to space $X \times Y$ and a membership function $\mu_R : X \times Y \to [0, 1]$. For $x \in X$ and $y \in Y$, $\mu_R(x, y)$ is interpreted as the degree of membership of (x, y) in R. In addition, an n-ary fuzzy set may be defined in the space $X_1 \times X_2 \times \cdots \times X_n$ with the membership function $\mu_R(x_1, x_2, \cdots, x_n)$, where $x_i \in X_i$, $i = 1, 2, \cdots, n$.

A fuzzy set $\widetilde{F}$ in the space Ω can also be defined by the triplet $(\omega, \widetilde{A}, \mu_F)$ (Dubois and Prade, 1992), where:

- ω is a subset of Ω.
- $\widetilde{A}$ is a linguistic label characterizing qualitatively part of the values of Ω.
- μ_F is the degree of membership of ω in the fuzzy set $\widetilde{F}$.

2.4.2 Possibility and Probability

Possibility measures (Dubois and Prade, 1988) can be recognized as one point of view on a fuzzy set. It is defined as:

$$\forall A, B, \Pi(A \cup B) = \max(\Pi(A), \Pi(B)) \tag{2.6}$$

The sets A and B in the possibility measures can be disjoint. It shows that when concerning about the dis-junctions of the events, we choose the maximum value of the event as the possibility. For the finite set Ω, we can also define the possibility measures on the singletons of Ω:

$$\forall A \Pi(A) = \sup\{\pi(\omega)|\omega \in A\} \tag{2.7}$$

Where $\pi(\omega) = \Pi(\{\omega\})$, ω is one set of the sets A; π is a mapping of Ω into $[0, 1]$ called a possibility distribution. If we consider about both ω from the space Ω and the membership function $\pi(\omega)$, it will refer to a fuzzy set.

However, when we talk about possibility, we will also consider about probability. A probability measures (Dubois et al., 2004) which is based on the occurrence of events can be defined as:

$$\forall A, \forall B, with A \cap B = \varnothing, P(A \cup B) = P(A) + P(B) \tag{2.8}$$

The sets A and B in the probability measures are disjoint. It shows that probability of an event is the percentage of the possibility from all the events and the sum of the probability of all the events is equal to 1. When we compare between the possibility and probability, possibility is seen as more flexible to deal with the relationship to the real system.

2.4.3 Fuzzy Control and Related Works on DEVS

Fuzzy control (Zimmermann, 1996) is able to control a fuzzy system with the help of human expertise. By using fuzzy control, it is able to construct a fuzzy inference systems (FIS). This system basically consists of fuzzification, defuzzification and fuzzy rules. FIS has been used to address the issue of incomplete knowledge in complex systems modeling. Bisgambiglia et al. (Bisgambiglia et al., 2008) present fuzzy modeling in a simple way to define complex system DEVS. They use interval or linguistic variables to make simulation possible. A library of fuzzy functions was added to DEVS formalism. Later Bisgambiglia et al. (Bisgambiglia et al., 2010) propose FIS with DEVS formalism in order to perform the control or the learning on systems described incompletely or with linguistic data. Santucci and Capocchi (Santucci and Capocchi, 2014) propose an approach based on the use of Fuzzy Control Language allowing facilitating the modeling and simulation of DEVS.

Fuzzy if-then rule (Chen and Tsao, 1989) (fuzzy rule, fuzzy implication or fuzzy conditional statement) as one of the fuzzy rules is defined that if x is A then y is B where A and B are linguistic values defined by fuzzy sets. Youcef and Maamar (Youcef and Maamar, 2014) use if-then rule to fuzzy reasoning rules obtained from observers or expert knowledge and specify a Fuzzy-DEVS model which computes this duration. They apply the method on forest fire propagation in the simulator to specify the new value in the model.

A fuzzy time control (Khan, 2008) is proposed for the time function in the discrete event systems. A fuzzy inference compositional rule is identified as the main theory for fuzzy time controller. This rule is defined as follows:

$$\mu_B(y) = \vee_y[\mu_A(x) \wedge \mu_F(x, y)] \tag{2.9}$$

A is a fuzzy set of input and F is a binary fuzzy set on space $X \times Y$. $x \in X$ and $y \in Y$, $\wedge$ means min and $\vee$ means max. According to the if-then rule, we need the inference of the input A and the implication $A \rightarrow B$ to determine the output B. We can consider B as the fuzzy set of time and the result can be inferred from input

variable and fuzzy time control. Besides, Giambiasi et al. (Giambiasi et al., 1994) also propose logic gates with fuzzy delays for modeling and simulation.

Genetic algorithm (Mitchell, 1998) can also be used as a fuzzy rule in FIS. Zeigler et al. (Zeigler et al., 1996) propose to integrate genetic algorithm and fuzzy inference system with DEVS. This genetic algorithm can be extended by a multilevel resolution search strategy in order to solve different degrees of abstracted problems (Kim and Zeigler, 1996).

2.4.4 Defuzzification Methods and Related Works on DEVS

The main idea of defuzzification is to change fuzzy sets into crisp sets. Centroid Method (Ross, 2009) is identified as the most prevalent and intuitively appealing of all the defuzzification methods. It is also called center of gravity or center of area. It is described by the following equation:

$$u = \frac{\sum_{i=1}^{n} \alpha_i M_i}{\sum_{i=1}^{n} \alpha_i A_i} \tag{2.10}$$

Where i is the rule of fuzzy sets, M_i is the membership function, A_i is the corresponding area and α_i is the degree that the rule i is fired. In the case of a continuous space, this method is described by:

$$u = \frac{\int_U u\mu_U(u)du}{\int_U \mu_U(u)du} \tag{2.11}$$

Moreover, the centroid method can be extended to center of largest area. If the output fuzzy set has at least two convex sub-regions, we select the largest area to centroid method.

Center of sum (Lee, 1990) is quite similar to the centroid technique but computationally more efficient. The difference between these two techniques is that the overlapping area is not merged in center of sum. Center of sum is described as follows:

$$u = \frac{\sum_{i=1}^{l} u_i \cdot \sum_{l=1}^{n} \mu_k(u_i)}{\sum_{i=1}^{l} \sum_{l=1}^{n} \mu_k(u_i)} \tag{2.12}$$

Besides, defuzzification methods also include max-membership principal which is similar to possibility measures as explained in equations 2.6 and 2.7. Weight average method is calculated by the output of each functions and their maximum membership

functions. This method is used in the Adapted Fuzzy Time Controller in section 4.2.3 and explained in the equation 4.10. Max-mean membership is to calculate the mean value which has the maximum membership function. First (or last) of maxima is calculated by the smallest value which has the maximum membership function.

Bisgambiglia et al. (Bisgambiglia et al., 2008) propose to use Expected Existence Measures method (EEM) in the DEVS atomic model. EEM is set based on fuzzy interval. It enables to add to this deffuzification technique a coefficient decision support. Based on different coefficient, the model can return a crisp time in different conditions.

2.4.5 Fuzzy Cluster

Since we focus on fuzzy cluster in this thesis, we give a general introduction of cluster analysis. The aim of cluster analysis is finding groups in data (Kaufman and Rousseeuw, 2009). The clusters can be subsets, groups or classes. Cluster analysis is often recognized as a branch of pattern recognition and artificial intelligence because finding objects into groups is an important human behavior. For example, a child needs to distinguish between men and women, between beds and chairs. Moreover, cluster analysis is able to impose a structure on a homogeneous data set for example dividing a country into telephone areas. The partition of clusters should have the following properties:

- Homogeneity within the clusters, i.e. data which belongs to the same cluster should be as similar as possible.
- Heterogeneity between the clusters, i.e. data which belongs to the different clusters should be as different as possible.

The clusters are not defined in advance. In the past, they are defined based on expert assumptions and judgments. For example in Figure 2.8, there are 11 objects in the data space. These objects may describe geography, medicine, chemistry and so on. By using human eye-brain system, we can observe two main clusters and two intermediate points. However, this kind of partitioning is not based on objectivity standards of modern science and there is a need to classify objects in more than two clusters. Over the last 40 years, a big amount of algorithms has been developed for cluster analysis. The main families of conventional clustering techniques include incomplete or heuristic cluster analysis techniques, deterministic crisp cluster analysis

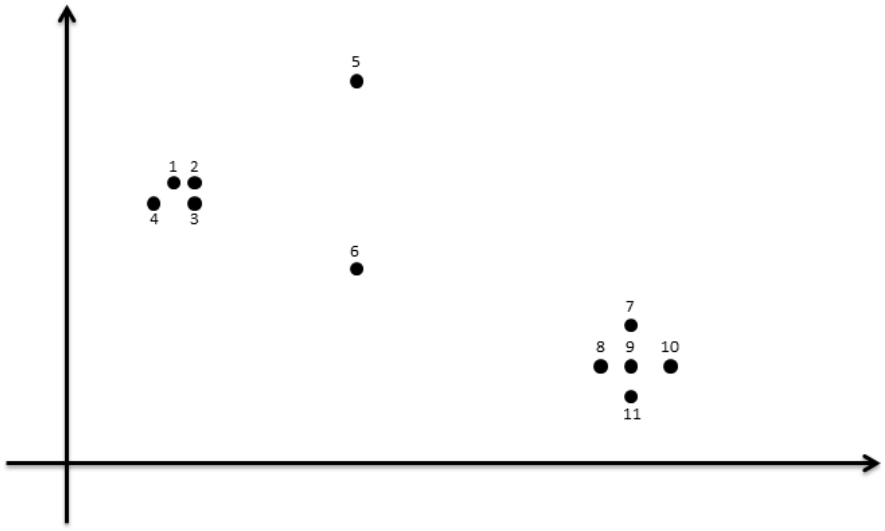


Figure 2.8: An example of cluster analysis.

techniques, overlapping crisp cluster analysis techniques, probabilistic cluster analysis techniques, possibilistic cluster analysis techniques, hierarchical cluster analysis, objective function based cluster analysis techniques, cluster estimation techniques.

k-means clustering is used in the process mining (Van der Aalst, 2011). The main idea is to divide the instances into several homogeneous groups. For this reason, we can get smaller data-sets and apply the additional process mining techniques. k represents the number of the clusters. In the k-means, we first set the centroid which is positioned randomly or regularly. Then we assign every instance to the closest centroid. Once we get the corresponding clusters, we set the new position of the centroid. Again and again, we assign every instance to the closest centroid and set the new position of the centroid until nothing changes and the position of the centroid remains the same. At last, the clusters are obtained.

Besides the clustering techniques above, fuzzy cluster (Höppner, 1999) applies fuzzy set logic into cluster analysis to represent the imprecision. The reason of choosing fuzzy cluster is that a deterministic cluster makes a hard partition of the data set. Fuzzy cluster allows for some ambiguity in the data. Each data set can belong to various clusters with a corresponding possibility. This possibility is quantified by means of membership coefficients which range from 0 to 1. For example in Figure 2.8, imagine that we apply the deterministic cluster and ask for two clusters. In that case, the program needs to make a decision to put object 5 to the cluster 1, 2, 3, 4 or to the cluster 7, 8, 9, 10, 11 since object 5 lies almost the same distance from both. Fuzzy cluster can provide a better decision for this situation when asking for two

clusters. Objects 1, 2, 3, 4 which have strong relationship with cluster 1 can have high membership coefficient (0.9) to cluster 1 and low membership coefficient (0.1) to cluster 2. Objects 7, 8, 9, 10, 11 which are strongly associated with cluster 2 can have high membership coefficient (0.9) to cluster 2 and low membership coefficient (0.1) to cluster 1. Object 5 may belong for the membership coefficient of 0.7 to cluster 1, for the membership coefficient of 0.2 to cluster 2. Object 6 may belong for the membership coefficient of 0.6 to cluster 1, for the membership coefficient of 0.3 to cluster 2. In addition, we can get a list of membership coefficient between objects and clusters. On the one hand, the main advantage of fuzzy cluster is to provide more detailed information of the data. On the other hand, the disadvantage is that as the number of objects becomes important, the amount of output information increases very fast and it reduces the performance of data analysis.

2.5 Conclusion

Theory of modeling and simulation provides detailed description about specifications and formalisms to represent systems. Fuzzy-DEVS is selected as the simulation model in this thesis and Fuzzy-DEVS considers about the imprecision to represent event data. The graphical notation helps to present the internal transition and external transition in the visualization. SES can extend the semantic aspects of Fuzzy-DEVS models. However, most of the studies on Fuzzy-DEVS has a limitation that the model does not come from real data. Process mining provides some potential methods which start from event logs and try to mine a process model. TS is identified as one of the most basic process models. In chapter 3, we will explain how to transform event data to event logs.

There are many different process discovery techniques. α algorithm provides the most basic process mining technique. Compared with α algorithm, heuristic mining is able to discover causality from event logs and region-based mining can capture complex process patterns. Region-based mining, also called Two Phase Approach, is reused to transform event logs to TS in chapter 3. Genetic mining improves model in a random way but it is difficult to realize on real data.

The basic concept in fuzzy logic is fuzzy sets. Fuzzy control can help to deal with time information in chapter 4. In the fuzzy control system, the implementation of weight average method is easier than the other deffuzification methods. When we apply fuzzy sets in the cluster analysis, we can get fuzzy cluster. Fuzzy cluster helps to mine coupled model from event data in chapter 4.

Chapter 3

Extracting Event Logs and Transition System From Event Data

3.1 Introduction

As soon as the event logs are available, process mining will be ready to execute. The event logs are the starting point of process mining. It has been shown that process mining can audit relevant information from event logs (Jans et al., 2014). A conceptual approach is proposed in (Van der Aalst, 2015) to extract event data from databases. Both studies provide general and abstract methods without results from real case study. For this reason, we choose event data as the input side of D2FD method. A rather loose definition and 16 guidelines of event data are shown in section 3.2. Here, a toy case of an e-shopping company is used to support the whole presentation in this chapter. A five-steps method is proposed to transform event data to event logs in section 3.3. Two Phase Approach in the process mining is reused in the D2FD method to transform event logs into TS in section 3.4.

3.2 Background

The starting point of D2FD method is event data observed from the real world. The event data is recorded based on some conditions. In this section, we give the definition of event data as well as sixteen guidelines. In addition, we introduce the toy case study which we use to support the explanation in chapter 3 and 4.

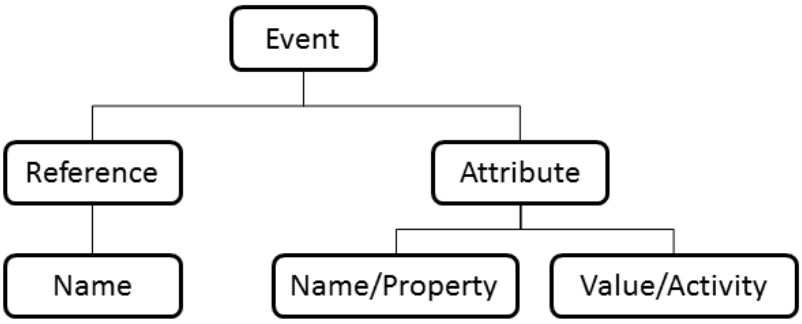


Figure 3.1: Structure of the event in event data.

3.2.1 Definition of Event Data

In Figure 1.2, the real world consists of a lot of things for example people, machines, organizations, business processes and so on. This world refers to the source level in Figure 1.1. Information systems can be controlled or supported by the real world. Then event data is recorded in the information system. The focus of this thesis is on the output side of the world, i.e., event data. Events can be of various types, and they can be recorded in various ways. Here, we give a rather loose definition of event data: event data is made up of events which describe the things happening. The general structure of the event is presented in Figure 3.1. Events are described by the references and attributes. Reference has a name and refers to some object for example person, machine, product and so on. Attribute has a name and a value. Sometimes the value can be the activity. Attributes can contain many perspectives like time, age, function, category and so on. Properties make identifiers for the references and attributes. Properties are quite like a conceptual classification for all the variables. The name of attributes can be the properties. When the event data is not stored only in one document, the documents can be classified as start document, middle document and end document. One principle of classification of the documents is based on time information. The one with earlier time is selected as start document. Another principle of classification is based on the relationship between documents. The one with higher level properties is selected as start document. For example, the document which records the purchase information of products is the start document. The document with the purchase request is the middle document and the document with invoice is the end document.

3.2.2 16 Guidelines of Event Data

Based on the concepts of event data we define 16 guidelines. Among the 16 guidelines, the first 12 guidelines come from the work of Van der Aalst (Van der Aalst, 2015) and we propose four more guidelines for a proper handling of event data. The 16 guidelines are shown as follows:

1. Reference and variable names should have clear semantics, i.e., they should have the same meaning for all people involved in creating and analyzing event data. Different stakeholders should interpret event data in the same way.

2. There should be a structured and managed collection of reference and variable names.

3. References should be stable (e.g., identifiers should not be reused or rely on the context). For example, references should not be time, region, or language dependent.

4. Attribute values should be as precise as possible. If the value does not have the desired precision, this should be indicated explicitly (e.g., through a qualifier). For example, if for some events only the date is known but not the exact time-stamp, then this should be stated explicitly.

5. Uncertainty with respect to the occurrence of the event or its references or attributes should be captured through appropriate qualifiers. For example, due to communication errors, some values may be less reliable than usual.

6. Events should be at least partially ordered. The ordering of events may be stored explicitly (e.g., using a list) or implicitly through a variable denoting the time-stamp of event.

7. If possible, also store transactional information about the event (start, complete, abort, schedule, assign, suspend, resume, withdraw, etc.). It is recommended to store activity references to be able to relate events belonging to the same activity instance.

8. Perform regularly automated consistency and correctness checks to ensure the syntactical correctness of the event log.

9. Ensure comparability of event logs over time and different groups of cases or process variants.

10. Do not aggregate events in the event log used as input for the analysis process.

11. Do not remove events and ensure provenance. Reproducibility is key for process mining.

12. Ensure privacy without losing meaningful correlations.

13. Event data at least needs to be recorded in csv or excel files.

14. The value names should be simple, precise and clear. Similar names with similar meanings can be found in different documents. A new value name can only be added if there is consensus on its meaning.

15. In the attributes, the start time and the finish time in the properties are mandatory. Each event refers to a case and instance defines a specific sequence of case. References are often identified as a label of instance.

16. Events are ordered, firstly by instance, and secondly increasingly by start time.

The above guidelines are very general and organized for D2FD method. The main purpose is to set a threshold of the input of D2FD method.

3.2.3 Toy Case Study

The toy case study is about an e-shopping company Peri. This company would like to know its business process and improve it. This toy case study will support the presentation of D2FD method in chapter 3 and 4. The event data of this toy case is shown in Table 3.1 and Table 3.2. According to the relationship, the start document records company request and the end document records customer orders. In Table 3.1, the e-shopping products indicate references (X_1 is the name of the product). One product can have several orders (X_1 has order 1, 2, 3 and 4). Attributes have the start time (00 : 00), the end time (00 : 10), the e-shopping shops (Peri), the product departments (Clothing in the PD field) and the product subthemes (Women in the PS field). Product, Order, Start Time, End Time, Shop, PD, PS are the properties. In Table 3.2, the customers indicate references (Y_1 is the name of the customer). One customer can place several orders (Y_1 has order 1 and 2). Attributes have the start time (03 : 40), the end time (03 : 50), the e-shopping shops (Peri) and the product departments (Electronic in the PD field). Customer, Order, Start Time, End Time, Shop, PD are the properties.

Table 3.1: Start document of toy case

Product	Order	Start Time	End Time	Shop	PD	PS
X_1	1	00:00	00:10	Peri	Clothing	Women
X_1	2	00:20	00:30	Peri	Clothing	Men
X_1	3	00:40	00:50	Peri	Clothing	Luggage
X_1	4	01:00	01:10	Peri	Electronic	TV & video
X_2	1	01:20	01:30	Peri	Clothing	Women
X_2	2	01:40	01:50	Peri	Clothing	Luggage
X_2	3	02:00	02:10	Peri	Clothing	Men
X_2	4	02:20	02:30	Peri	Electronic	TV & video
X_3	1	02:40	02:50	Peri	Clothing	Women
X_3	2	03:00	03:10	Peri	Electronic	Computer
X_3	3	03:20	03:30	Peri	Electronic	TV & video

Table 3.2: End document of toy case

Customer	Order	Start Time	End Time	Shop	PD
Y_1	1	03:40	03:50	Peri	Sports
Y_1	2	04:00	04:10	Peri	Electronic
Y_2	1	04:20	04:30	Peri	Electronic
Y_2	2	04:30	04:50	Peri	Sports

According to guideline 14, the e-shopping shop (Peri) is similar in both documents as well as the Electronic in PD. According to guideline 15, in Table 3.1, the e-shopping products indicate instances (X_1). One product has several orders, each being a case at a different time (start time and end time). In Table 3.2, the customers are the instances (Y_1). One customer has several orders, each being a case at a different time (start time and end time). According to guideline 16, in the same instance, the events are ordered by start time. For example the order 1 of X_1 is smaller than the order 2.

3.3 Five-Steps From Event Data to Event Logs

The event data has a very big amount which takes time to analysis them. So we would like to focus on the important part of the event data. For this reason, we obtain the goals by making interviews with business people and organizations. In addition, we construct the conceptual structure to identify the underlying relationships, classify the values between public values and private values and select the process instance

based on properties and documents. At last, the elements of event data are converted into the elements of event logs.

3.3.1 Setting up of Goals

Prior to any other steps in the five-steps method is the definition of goals. Goals are investigated from interview. They include the problem to be addressed or the performance to evaluate. The goals can guide the process of D2FD method until the validation of D2FD method. In the toy case, two goals are identified:

- how does the e-shopping company work?
- how does the customer choose products?

3.3.2 Identification of Relationships

One of the challenge in section 1.2.1 is event correlation. Raw data are not often well organized, and there is more than one way to build event logs that can store them. Identifying underlying relationships can help researchers to find analysis results. The references in the event data may not have relationships in that they are unique and special. However, the values in the attributes can have different relationships between each other. Here, we propose a conceptual structure to represent the relationships of the values. This conceptual structure is inherited from SES which keeps the function of aspect, multi-aspect, specialization and variable. The reason to choose SES is because SES has more functions to represent hierarchical system which is better than Unified Model Language (UML). In SES, aspects represent ways of decomposing entities into more detailed parts. Multi-aspects are aspects for which the components are all of one kind. Specializations represent categories in specific forms that an entity can assume. The difference between the new conceptual structure and SES is that entity is replaced by value and property defines each level of the structure. When the event data has more than one document, we propose that each document generates one conceptual structure as well as one Fuzzy-DEVS atomic model. On the one hand, the identification of relationships can be obtained from business interview. On the another hand, the construction of the conceptual structure can be done by domain experts.

In the toy case, two conceptual structures are constructed as shown in Figure 3.2. There are three properties (shop, product departments and product subthemes) pointing out three levels of the structure. Here, we use P, CL, E, S, W, M, L, T and CO

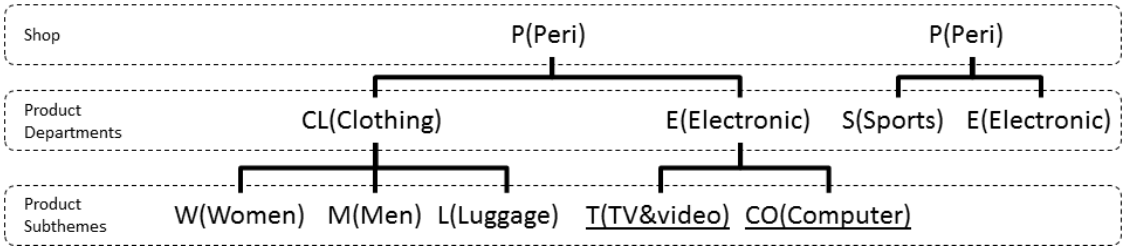


Figure 3.2: The two conceptual structure of the toy case.

to represent the values in the attributes Peri, Clothing, Electronic, Sports, Women, Men, Luggage, TV&video and Computer. These values come from the elements in Tables 3.1 and 3.2. In Table 3.1, CL and E are the aspects of P, while W, M, and L are the aspects of CL, and T and CO are the aspects of E. Respectively, in Table 3.2, S and E are the aspects of P.

3.3.3 Identification of Values

Once the relationships of the values exist in one document, we use aspect, multi-aspect, specialization and variable to describe them in the conceptual structure. When the values relate to the ones in another document, we need to distinguish public and private value. Hence, the relationship in this situation is similarity. The rules of identifying public and private value can help to find modularity between documents. The rules are shown as follows:

- If one value has a strong relationship with the value in another document, this value is identified as public value and the children of this value as private value.
- Values which do not have relationships with any other ones, are identified as public values.

In the toy case in Figure 3.2, the value E in the start document is similar in the end document. There is a connection in the level of product departments. Therefore, the children of E, i.e., T and CO, are private values, underlined in the Figure 3.2. P, CL, E, S, W, M and L are public values.

3.3.4 Selection of Process Instance

The process instance (Mieke, 2015) is the object that one follows throughout the business process. There are two dimensions of process instance: start, middle and end document; property level of the conceptual level. In a document-based business

process, the process instance is related to at least one document. In some cases, all relevant knowledge resides in a single document (e.g., the start document). In other cases, the knowledge spreads among various documents. If there is only one single entry of the process, we choose start document. If there are multiple entries of the process, we choose start document with the purpose of efficiency and end document with the purpose of compliance.

Based on the conceptual structure in Figure 3.2, there are different property levels. The problem happens when the value can be either public value or private value. For example, CL and E are possible to become private value in that P is similar in two documents. For this reason, we propose to select one level (or we can say one property) which involves the key and interesting values. In case of mixed levels, we take the higher level as the process instance.

In the toy case, both the start document and the end document are involved in the process instance selection. The key values are the leaves of the conceptual structures built (i.e., {W, M, L, T, CO} at one side, and {S, E} at the other side).

3.3.5 Mapping Between Event Data and Event Logs

In Figure 3.3, a general mapping from event data to event logs is proposed to convert instance to trace, case to event id, time to time extension, and values to concept extension. The values are selected from the conceptual structure in the property level. In the toy case, Women, Men, Luggage, TV&video, Computer are selected as final values in the start document and Sports, Electronic are selected as final values in the end document. The first event logs of the start document is shown as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<log xes.version="1.0" xes.features="nested-attributes
    " openxes.version="1.0RC7">
    <extension name="Time" prefix="time" uri="http://
        www.xes-standard.org/time.xesext" />
    <extension name="Concept" prefix="concept" uri="
        http://www.xes-standard.org/concept.xesext" />
    <classifier name="Event Name" keys="concept:name" />
    <string key="concept:name" value="Start document.
        csv" />
    <trace>
        <string key="concept:name" value="X1" />
        <event>
            <string key="End time" value="00:10" />
```

```
            <string key="concept:name" value="Women"/>
            <string key="time:timestamp" value="00:00"/>
        </event>
        <event>
            <string key="End time" value="00:30"/>
            <string key="concept:name" value="Men"/>
            <string key="time:timestamp" value="00:20"/>
        </event>
        <event>
            <string key="End time" value="00:50"/>
            <string key="concept:name" value="Luggage"/>
            <string key="time:timestamp" value="00:40"/>
        </event>
        <event>
            <string key="End time" value="01:10"/>
            <string key="concept:name" value="TV&Video"/>
            <string key="time:timestamp" value="01:00"/>
        </event>
    </trace>
</log>
```

The event log above has only one log using XES version 1.0. All the attributes are nested attributes. There are the extension of concept and time and one classifier. This event log is constructed from "Start document.csv". Here, we only show the first trace X_1 with four events. Each event has time-stamp which contains start time (00 : 00), concept:name in the property of PS (Women) and End time (00 : 10). Correspondingly, the second event logs of the end document is shown as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<log xes.version="1.0" xes.features="nested-attributes
    " openxes.version="1.0RC7">
    <extension name="Time" prefix="time" uri="http://
        www.xes-standard.org/time.xesext"/>
    <extension name="Concept" prefix="concept" uri="
        http://www.xes-standard.org/concept.xesext"/>
    <classifier name="Event Name" keys="concept:name"/>
    <string key="concept:name" value="End document.csv"
        />
    <trace>
        <string key="concept:name" value="Y1"/>
        <event>
            <string key="End time" value="03:50"/>
            <string key="concept:name" value="Sports"/>
            <string key="time:timestamp" value="03:40"/>
```

```
        </event>
        <event>
            <string key="End time" value="04:10"/>
            <string key="concept:name" value="Electronic"
                />
            <string key="time:timestamp" value="04:00"/>
        </event>
    </trace>
</log>
```

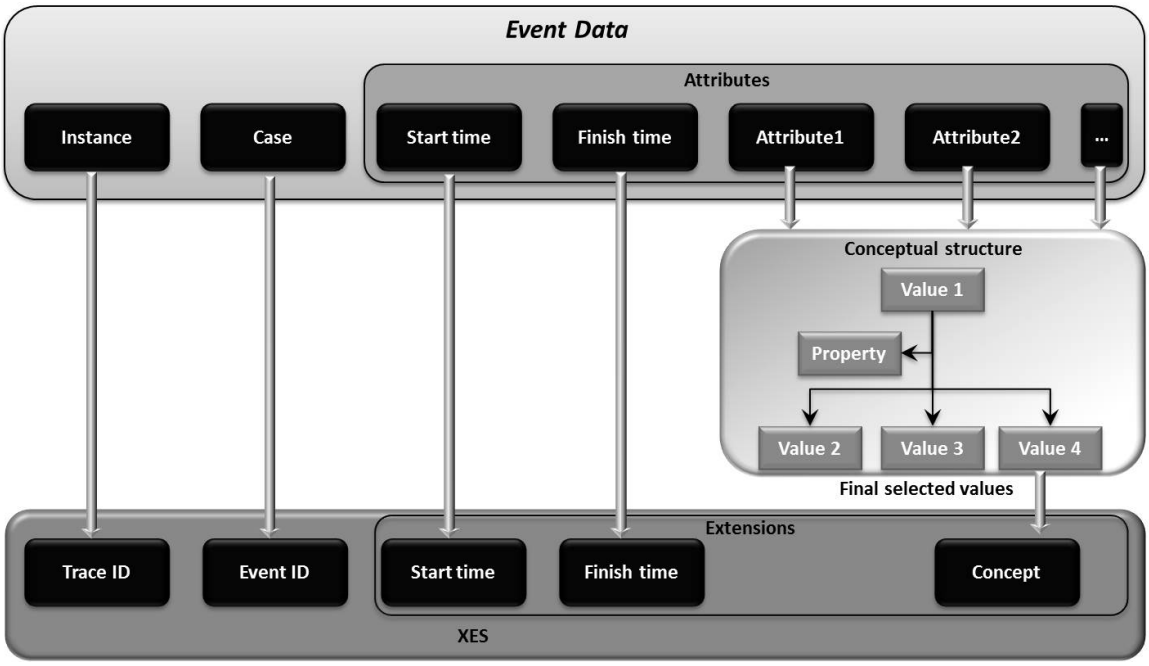


Figure 3.3: General mapping between event data, conceptual structure and XES.

The event log above is constructed from "End document.csv". It uses XES version 1.0 and all the attributes are nested attributes. There are the extension of concept and time and on classifier. Here, we only show the first trace Y_1 with four events. Each event has time-stamp which contains start time (03 : 40), concept:name in the property of PS (Sports) and End time (03 : 50).

3.4 From Event Logs to Transition System

Two Phase Approach in process mining (Van der Aalst, 2011) provides a discovery technique which first transforms an event log into a low-level TS and then synthesizes a Petri net from TS. The D2FD approach reuses the first stage of the

Two Phase Approach (i.e., the production of a TS). The second stage of the Two Phase Approach is explained in section 2.3.3.2.

In order to construct a TS, we need to determine the set of states. Every position in a trace of an event log corresponds to a state of a TS. In the toy case, the event log L_{start} is extracted from start document with three traces and the event log L_{end} is extracted from end document with two traces. Event log $L_{start} = [(W, M, L, T), (W, L, M, T), (W, CO, T)]$. Event log $L_{end} = [(S, E), (E, S)]$. We take the first trace in L_{start} as the example (W, M, L, T). When the current state is between M and L, the partial trace $\sigma_{past} = W, M$ describes the past of the corresponding case and the partial trace $\sigma_{future} = L, T$ describes the future of the corresponding case.

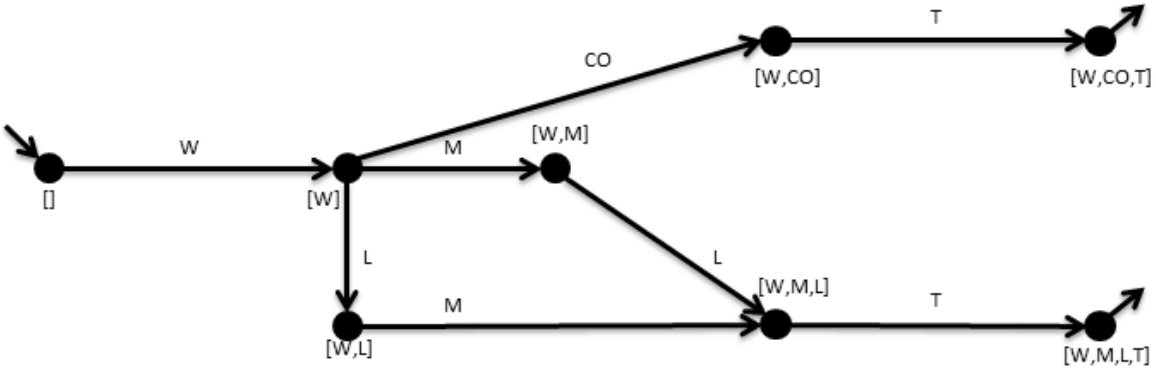


Figure 3.4: TS model from the start document of the toy case.

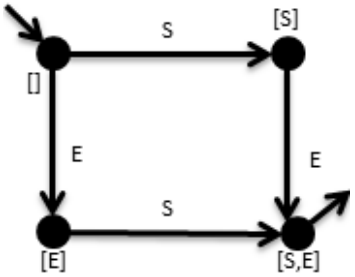


Figure 3.5: TS model from the end document of the toy case.

Different methods are used to capture states, i.e., past or future; set abstraction or multi-sets abstraction; k-tail method. Past or future means both order and frequency matter. Set abstraction means neither order nor frequency matter. Multi-sets abstraction means only frequency matters. K-tail method proposes to focus on the last one or more activities to capture state or focus on the next one or more activities to capture state. By organizing different methods, we can get different kinds of TSs.

In the toy case, the state is represented by the multi-sets abstraction in the start document. The first three activities (W, M, L), (W, L, M), (W, CO, T) are considered as the multi-sets. Then they add T at the end of first two traces. The TS from the start document is shown in Figure 3.4. The label of each state has all the activities instead of the process of all the activities for example $[W, M, L]$, it has three activities but there are two processes (W, M, L) and (W, L, M). The transition is represented by arc and the label of the transition shows the activity. There are one initial state ([ ]) and two final state ($[W, M, L, T]$ and $[W, CO, T]$). In the end document, the state is represented by the set abstraction. The TS from the end document is shown in Figure 3.5. There is one initial state ([ ]) and one final state ($[S, E]$).

3.5 Conclusion

In this chapter, we propose a five-steps method to extract event logs from event data. This method tries to solve the challenge in section 1.2.1. The event data is defined at first as the input side of D2FD method. Four more guidelines are proposed to improve the efficiency and quality of mining event data. We extract event logs from data source like CSV file. Once the event data is observed, goals are investigated from the business interview. Some underlying relationships can be observed between values. The conceptual framework is used to build a modular and hierarchical abstraction from the data collected, in order to structure the event log that will be extracted. The event correlation is discovered by the conceptual framework. The identification of public value and private value not only makes connections between documents, but also observes the modularity to construct internal transitions and external transitions in DEVS. In order to reduce the scope of event data, we put the emphasis on the interesting values, properties and documents. At the end, a corresponding mapping realizes the transformation from event data to event logs. Two Phase Approach helps to transform event logs to TS. Two TS models are obtained in Figures 3.4 and 3.5 in the toy case study. In the following chapter, as the main part of D2FD method, we continue to construct a Fuzzy-DEVS model from TS.

Chapter 4

Mining Fuzzy-DEVS Model From Transition System

4.1 Introduction

In the process mining, the resulting process model is usually a Petri net model (Peterson, 1977). Two techniques exist to discover the Petri net model: the α-algorithm and the Two Phase Approach. The α-algorithm is able to discover concurrency but is unable to take frequencies into account. Two Phase Approach first learns a TS from event logs and then uses region-based approach to construct a Petri net from TS. A major drawback of this approach is that the discovered Petri net cannot represent the timing aspects. Also, there is a lack of modularity, making the design of hierarchical models difficult to realize. Therefore, process mining shows limitations in inferring complex systems. D2FD method is proposed to overcome these limitations, which is rooted in the system-theoretic power of the DEVS formalism. Compare with Petri net, DEVS is able to provide more accurate model due to the following facts:

- DEVS gives a more general framework for modeling and simulation of complex systems.
- DEVS integrates naturally the notion of time contrary to Petri nets which require an extension of the formalism.
- DEVS offers a formal (and separated from model) definition of the simulator.

Fuzzy-DEVS adds to DEVS capabilities to take frequency of events and imprecise knowledge into account by applying fuzzy sets theory. Another candidate of the formalism for providing similar advantages is Stochastic DEVS model, yet Fuzzy-DEVS is more convenient than Stochastic DEVS for the following reasons:

- the concept of possibility used in Fuzzy-DEVS emphasizes the likelihood of an event in the system in a more objective manner than the concept of probability used in Stochastic DEVS.
- the concept of possibility allows users to focus only on the mainstream behavior of the business process.
- Fuzzy-DEVS can provide more semantics by integrating subjective data and linguistics.

Based on the general structure of D2FD method in Figure 1.2, we will explain the most important part from TS to Fuzzy-DEVS in section 4.2. The toy case of the e-shopping company in section 3.2.3 continues to be used in this chapter.

4.2 From Transition System to Fuzzy-DEVS Model

Since Fuzzy-DEVS model can be distinguished between atomic model and coupled model and DEVS can be extended to Fuzzy-DEVS, different methods are integrated together in the D2FD method. The transformation of all the functions in the original DEVS formalism is based on improved region-based approach. The core idea is to discover regions that correspond to states of Fuzzy-DEVS. A region is a set of states that all activities in the TS "agree" on the region. It can also detect the concurrency. All the behavior of the regions will be transformed to the state of DEVS. So from a big transition system, we can get a smaller Fuzzy-DEVS model. Dependency Method is used for producing the fuzzy internal transition, fuzzy external transition and fuzzy output function; AFTC is used for obtaining fuzzy time advance function. To execute the Fuzzy-DEVS model, Possibility Measures and the final output of AFTC are applied. The final output of AFTC is inferred by the weighted average method from defuzzification methods. Fuzzy cluster is applied for the functions of Fuzzy-DEVS coupled model.

4.2.1 Improved Region-Based Approach

After transforming the event logs into the low level process model TS, we can mine a Fuzzy-DEVS atomic model from it. In turn, this Fuzzy-DEVS atomic model can be coupled together to form the entire Fuzzy-DEVS model. According to the section 2.3.3.2, we propose an improved region-based approach to realize the transformation of Fuzzy-DEVS. Before starting introducing improved region-based approach,

we first define the concept of regions. The definition of regions is a set of states in the TSs based on some criteria. Let $TS = (S^T, A, T)$ be a TS where S^T is the set of the states, A is the set of the activities, T is the set of the transitions. P_a is the mean period time duration for each activity $a \in A$. R is a region and $R \subseteq S^T$ is a subset of the states. For each activity $a \in A$, there are four following criteria:

- All transitions $(s_1^T, a, s_2^T, P_a) \in T$ enter R and $P_{a_1} \in P_a$, $P_{a_2} \in P_a$, ..., $P_{a_n} \in P_a$, i.e. $s_1^T \notin R$ and $s_2^T \in R$ and $P_{a_1} \approx P_{a_2} \approx ... \approx P_{a_n}$.
- All transitions $(s_1^T, a, s_2^T) \in T$ exit R, i.e. $s_1^T \in R$ and $s_2^T \notin R$.
- All transitions $(s_1^T, a, s_2^T) \in T$ do not cross R, i.e. $s_1^T, s_2^T \in R$ or $s_1^T, s_2^T \notin R$.

Since there is no time function in the TS, more information are extracted from the event logs for example P_a is calculated by AFTC in section 4.2.3. For region R, there are three kinds of classification of the activities such as entering the region, leaving the region and non-crossing. The classification of non-crossing can be distinguished between the activity inside the region and outside the region. In other words, non-crossing activities always connect two states inside the region or outside the region. The conditions of the region can not be met when an activity enters in one side of this region and goes out in another side of this region. Since the region can have more than one activities, these activities need to have the approximated time duration each other. If these activities do not have approximated time, the regions will be split into smaller regions. Therefore, we are only interested in minimal regions. Figure 4.1 shows an example of defining regions. The dashed rectangle describes a region R which contains seven states in the TS. The label of the transition represents the activity so different transition may have same activity. All a-labeled transitions enter the region as well as b-labeled transition. If there is an a-labeled transition with two states outside or inside the region, R cannot be a region. All c-labeled transitions leave the region as well as d-labeled transition. All e-labeled transitions do not cross the region both inside the region and outside the region. The f-labeled transition and the g-labeled transition do not cross inside and outside the region respectively. Inside the region, the periods of activity P_a and P_b are approximated each other.

The advantage of this approach is to narrow a large TS into a smaller Fuzzy-DEVS model. However, the disadvantage of this approach is that there can be more than one way to demarcate the regions in the TS when the numbers of the states increase. This costs a lot to calculate in the computer system. For this reason, we propose a c-groups method where c is a performance coefficient. The coefficient c is

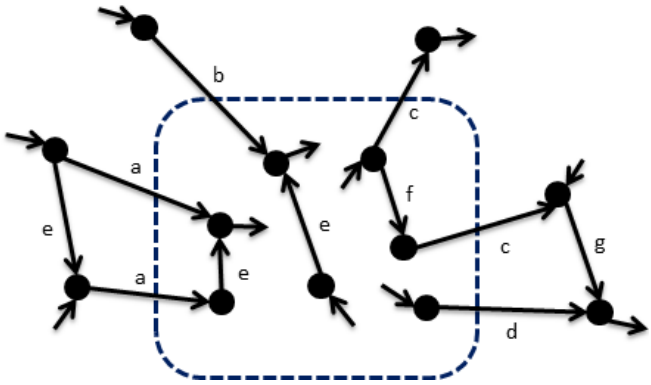


Figure 4.1: An example of improved region-based approach.

calculated based on the experiments on the computer. When testing the improved region-based approach, we choose the number of the states as much as possible in a TS model. If the time of computer operation is not over 30 minutes, we give this number of the states to c. Sometimes, c is smaller than the whole number of the states of TS. The group of the set of the states $\hat{S_i^T}$ in the TS is defined as follows:

$$\forall s^T \in S^T, \exists i = 0, 1, 2, ..., \hat{S_i^T} = \begin{cases} \{s^T_{g\times c}, s^T_{g\times c+1}, ..., s^T_{g\times c+c}\} & \text{for} \quad i < \frac{n}{c} \\ \{s^T_{g\times c}, s^T_{g\times c+1}, ..., s^T_n\} & \text{for} \quad i = \frac{n}{c} \end{cases} \tag{4.1}$$

Where n is the number of the states in the TS. The number $\frac{n}{c}$ represents the number of the states in each group. The advantage of this c-groups method is to improve the efficiency of the improved region-based approach. The shortcoming is that the state in one group and the state in another group cannot be put together in the same region. So we are not sure that the regions are the optimum by c-groups method.

The original region-based approach has an inability to discover particular process constructs. This can be handled through the extension which is so-called "forward closure" property. Once we have TS, we check certain properties. If this property does not hold, the labels need to be split. The activities are renamed into more values.

Once the regions are obtained, the functions of TS are transformed into the functions of Fuzzy-DEVS. Some information are extracted from event logs to complete this transformation. In the event logs, let pv be the private value and uv be the public value. This transformation follows the rules:

$$R \to S \tag{4.2}$$

Where the state of Fuzzy-DEVS atomic model $s \in S$.

$$uv \to x \bigcup y \tag{4.3}$$

Where the input value $x \in X$ and the output value $y \in Y$.

$$ta = \begin{cases} 0 & \exists s_0 \in S \\ T^F & \\ Infinite & \nexists S = S_1^I \end{cases} \tag{4.4}$$

Where s_0 is the initial state, T^F comes from AFTC in section 4.2.3, S_1^I is the input states of all internal transition.

$$\begin{gathered} T \rightarrow \widetilde{\delta}_{int} \\ (s_1^T, uv, s_2^T) \rightarrow (s_1, s_2, \mu_{int}) \end{gathered} \tag{4.5}$$

Where $s_1 \in R_1$ and $s_2 \in R_2$, μ_{int} comes from Dependency Method in section 4.2.2.

$$\widetilde{\lambda} : (y, \mu_{int}) \tag{4.6}$$

$$\begin{gathered} T \rightarrow \widetilde{\delta}_{ext} \\ (s_1^T, pv, s_2^T) \rightarrow (s_1, e, x, s_2, \mu_{ext}) \end{gathered} \tag{4.7}$$

Where the elapsed time $e : 0 \leqq e \leqq ta$, μ_{ext} comes from Dependency Method in section 4.2.2.

The regions are related to the states of Fuzzy-DEVS in equation 4.2. Public values uv in the event logs are always used as either input event or output event, as suggested by equation 4.3. In equation 4.4, the fuzzy time advance $\widetilde{ta}$ is defuzzed into a crisp value ta. If the state is the initial state, the time is 0. If the state does not belong to the input states of all internal transition (in another words, the state has no internal transition to the next state), the time is infinite. Otherwise, the time is calculated from AFTC in section 4.2.3. The transition in the TS is transformed into internal transition if the activity is public value illustrated in equation 4.5. The possibility of the output function μ_{int} in equation 4.6 is the same as the internal transition. μ_{int} is calculated by Dependency Method in section 4.2.2. In equation 4.7, the transition in the TS is transformed into external transition if the activity is private value. If the elapsed time e is smaller than time ta, this external transition can be executed. The possibility of the external transition μ_{ext} is different from μ_{int} but calculated from the same method.

To illustrate the improved region-based approach, we use the same toy case which starts from Table 3.1 and Table 3.2. Figure 4.2 shows how the transition system (upper level of the figure) of the toy case results in its corresponding Fuzzy-DEVS atomic model (lower level of the figure). This TS relates to the start document of the toy case and is same as Figure 3.4. The states of TS are split into six regions

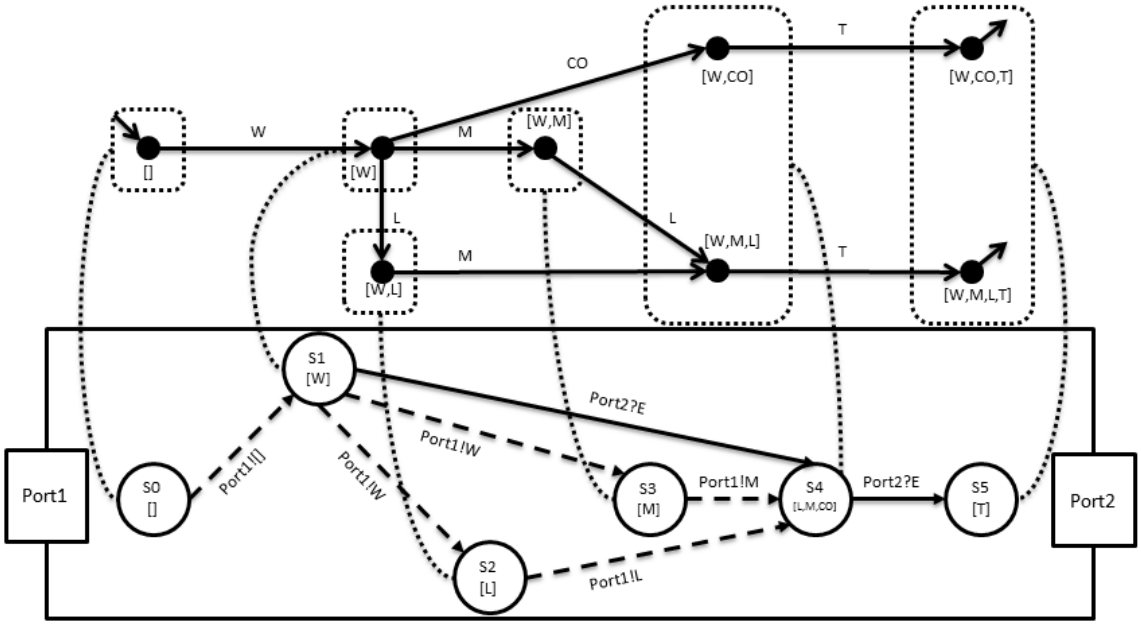


Figure 4.2: First toy case from TS to Fuzzy-DEVS atomic model.

because P_W, P_M, P_L are not approximated each other. The region which contains $[W, CO]$ and $[W, M, L]$ stays as P_{CO} is approximated. The states of Fuzzy-DEVS are converted from these six regions according to equation 4.2. The labels of the states in Fuzzy-DEVS are unique such as $S0$, $S1$, $S2$, $S3$, $S4$, $S5$. The activities in the label of the states show that the ones have approximated time duration. The initial state [ ] is transformed into $S0$. As we know, P, CL, E, S, W, M and L are public values. T and CO, are private values. According to equation 4.5 and 4.7, the transitions which contains W, M and L are converted to internal transition and the transitions which contain CO and T are converted into external transition. The labels of these transitions are using the graphical notation in section 2.2.2.2. There are `Port1` and `Port2` in the box of Fuzzy-DEVS box. As illustrated in equation 4.3, public values W, M, L represent the output events. The mother of private values CO and T, which is also public value E, represents the input event. The internal transition is depicted as dotted arc with the corresponding output event such as `Port1`!W, `Port1`!M, `Port1`!L. As $S0$ is the initial state, based on equation 4.4, the time ta_{S0} is 0. The internal transition is executed immediately with empty output event `Port1`![ ]. The external transition is depicted as solid arc with the corresponding input event `Port2`?E. Since $S4$ and $S5$ have no internal transition, the time ta_{S4} and ta_{S5} are infinite. The time ta_{S1}, ta_{S2} and ta_{S3} come from AFTC. The simulation of this Fuzzy-DEVS atomic model is based on its state, its transition and its time. For example, the state is $s1$

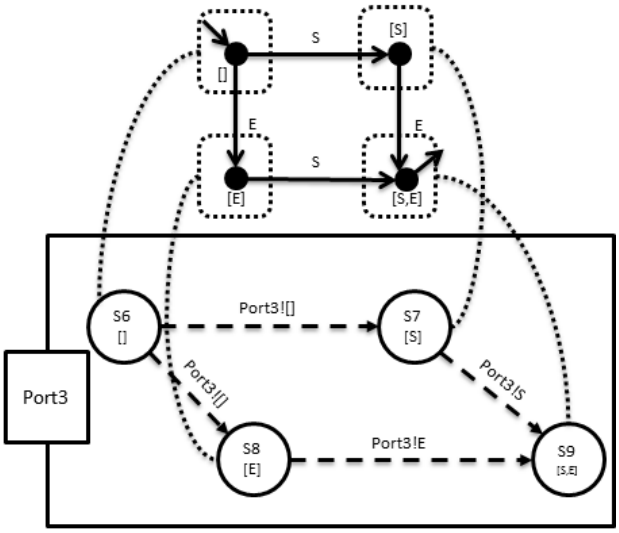


Figure 4.3: Second toy case from TS to Fuzzy-DEVS atomic model.

now. If the elapsed time e is less than ta_{S1} and the input event E comes from `Port2`, the external transition is executed. States change from $S1$ to $S4$. If the elapsed time e reaches ta_{S1}, the internal transition is executed. States change from $S1$ to either $S2$ or $S3$. The output event W is sent to `Port1`.

The TS is able to solve the problem of concurrency. The concurrency summarizes the situation that at least two activities are existing and occurring simultaneously or side by side. After transforming from TS, Fuzzy-DEVS is able to keep this concurrency. For example in Figure 4.2, the activities L and M are concurrent. The regions of these two activities are converted into $S2$ and $S3$ respectively. The states $S2$ and $S3$ are concurrent. Moreover, the internal transitions $\widetilde{\delta}_{int}(S1, S2)$ and $\widetilde{\delta}_{int}(S1, S3)$ have the corresponding possibilities. These possibilities provide more information than concurrency.

Figure 4.3 shows the second toy case from TS to Fuzzy-DEVS atomic model. This TS relates to the end document of the toy case and is same as Figure 3.5. Since P_S and P_E are not approximated each other, the four states of TS are split into four regions themselves then become the four states of Fuzzy-DEVS atomic model, i.e. $S6$, $S7$, $S8$, $S9$. $S6$ is the initial state. As S and E are public values, the transitions in TS are converted to internal transitions in Fuzzy-DEVS. Meanwhile, there is only one port `Port3`. The internal transition is depicted as dotted arc with the corresponding output event such as `Port3`!S and `Port3`!E. The time ta_{S6} is 0 and ta_{S9} is infinite. The other time ta_{S7} and ta_{S8} are calculated by AFTC. The state starts from $S6$ and executes the internal transition immediately. If the state is $S7$, the output event S

will send to `Port3` after ta_{S7}. If the state is $S8$, the output event E will send to `Port3` after ta_{S8}.

4.2.2 Dependency Method with Possibility Measures

Dependency Method in the heuristic mining in section 2.3.3.3 is used to get frequency between any pair of activities and the value of the dependency relation. We propose to use Dependency Method to generalize internal transition, external transition and output function into fuzzy sets. Since the transition and the activity are strongly linked together in TS, the frequency of the transition can obtain from the frequency of the activity. There are two steps in this method. The first step is to calculate the frequency of each transition from the instances of event logs. Then we use the following equation 4.8 to calculate the possibility of every transition.

$$\mu(S_i \to S_j) = \begin{cases} \left|\frac{F(S_i \to S_j) - F(S_j \to S_i)}{F(S_i \to S_j) + F(S_j \to S_i) + 1}\right| & \exists i \neq j \\ \frac{F(S_i \to S_j)}{F(S_i \to S_j) + 1} & i = j \end{cases} \tag{4.8}$$

In equation 4.8, μ and F respectively defines the possibility and the frequency of a transition from one state to another state. $\to$ means internal or external transition between two states, while i and j are the state numbers. The value of the dependency is between -1 and 1 in the heuristic mining. We take the absolute value which is between 0 and 1 to be able to apply in fuzzy sets. Since this value of the dependency comes from the frequency of the transitions in TS, it is given to the membership function of the internal transition and external transition in Fuzzy-DEVS. The possibility of the output function is designed to be equal to the possibility of the internal transition. In the graphical notation, we put this possibility at the end of the label of internal and external transitions. In the toy case, we can have a look at the control flow of the event logs. An event log can be converted to a multiset of traces and same trace may appear multiple times. Imagine that we get a longer set of event logs, for example there are two event logs $L_{start} = [(W, M, L, T)^{10}, (W, L, M, T)^{15}, (W, CO, T)^{3}]$ and $L_{end} = [(S, E)^{5}, (E, S)^{8}]$ where the numbers in exponent indicate the times of the corresponding traces. Table 4.1 presents the value of frequency and dependency of the model in Figure 4.2. Among them, the transitions $S1 \to S4$ and $S4 \to S5$ are external transitions and all other transitions are internal transitions. If the state has no internal transition or external transition, the possibility is 0 for example $\mu_{int}(S1 \to S4) = 0$, $\mu_{ext}(S0 \to S1) = 0$. The internal transition has the same possibility in output function for example $\mu_{int}(S0 \to S1) = 0.97$ is the possibility of $\widetilde{\lambda}(S0)$. Table 4.2 presents the value of frequency and dependency of the model

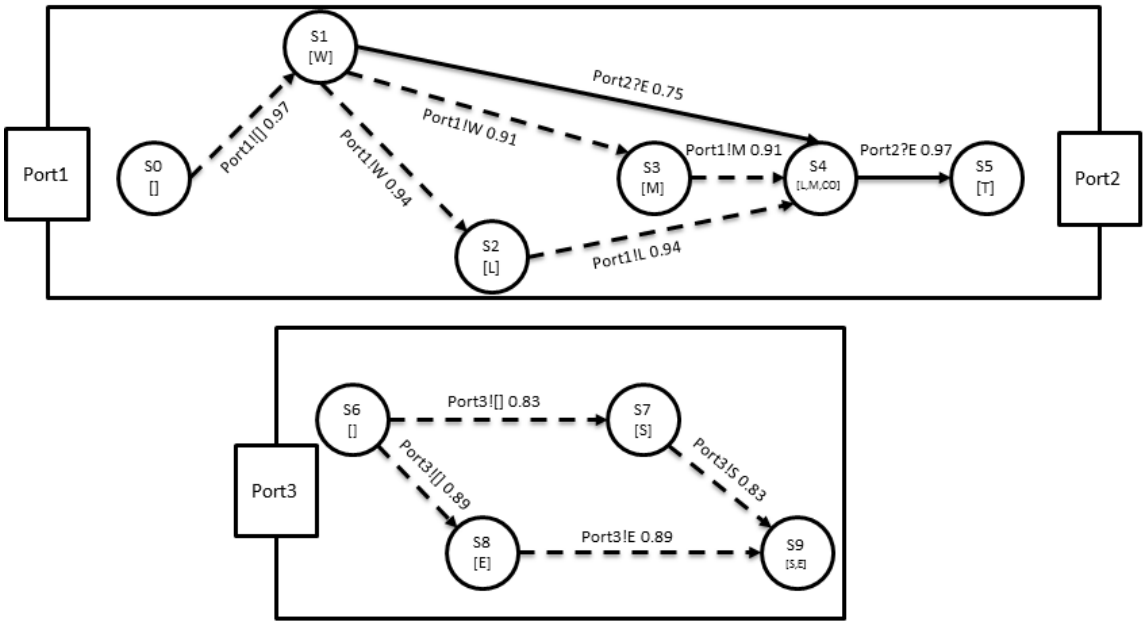


Figure 4.4: Two Fuzzy-DEVS atomic models using Dependency Method in the toy case.

in Figure 4.3. As all the transitions are internal transitions, all the possibilities are for internal transitions and output functions. All the possibilities in Table 4.1 and Table 4.2 are integrated with the graphical notation in Figure 4.4 for example $\widetilde{\delta}_{int}(S3, 0.91) = S4$, $\widetilde{\lambda}(S3, 0.91) = Port1!M$, $\widetilde{\delta}_{ext}(S1, Port2?E, 0.75) = S4$.

Table 4.1: Frequency and possibility of first toy case

	$S0 \to S1$	$S1 \to S2$	$S1 \to S3$	$S1 \to S4$	$S2 \to S4$	$S3 \to S4$	$S4 \to S5$
F	28	15	10	3	15	10	28
μ_{int}	0.97	0.94	0.91	0	0.94	0.91	0
μ_{ext}	0	0	0	0.75	0	0	0.97

Table 4.2: Frequency and possibility of second toy case

	$S6 \to S7$	$S6 \to S8$	$S7 \to S9$	$S8 \to S9$
F	5	8	5	8
μ_{int}	0.83	0.89	0.83	0.89

Possibility Measures are used to execute the Fuzzy-DEVS model for simulation because Possibility Measures can focus on the mainstream of the process. It is defined in section 2.4.2. This method is similar to a defuzzification method which changes

fuzzy sets into crisp value. When concerning about the disjunctions of the events, we choose the maximum value of the event as the possibility. For the internal transition and output function, if the elapsed time reaches ta, the one with the maximum possibility is chosen. For the external transition, if the elapsed time does not reach ta and input events X come, the one with the maximum possibility is chosen. In the toy case, for example the upper model in Figure 4.4, the state starts from $S0$ and goes immediately to $S1$ by $\widetilde{\delta}_{int}(S0, 0.97)$ and $\widetilde{\lambda}(S0, 0.97) = Port1![\ ]$. If the input event $Port2?E$ comes before ta_{S1}, the external transition is triggered and state goes to $S4$ (i.e. $\widetilde{\delta}_{ext}(S1, Port2?E, 0.75) = S4$). Otherwise, the internal transition $\widetilde{\delta}_{int}(S1, 0.94) = S2$ is executed with $\widetilde{\lambda}(S1, 0.94) = Port1!W$ as 0.94 is bigger than 0.91. Continually, after ta_{S2}, $\widetilde{\delta}_{int}(S2, 0.94) = S4$ is executed with $\widetilde{\lambda}(S1, 0.94) = Port1!L$. In the end, if the state is $S4$, it waits until $Port2?E$ comes to trigger the external transition $\widetilde{\delta}_{ext}(S4, Port2?E, 0.97) = S5$.

4.2.3 Adapted Fuzzy Time Controller (AFTC)

AFTC is developed from fuzzy discrete event controller system (FDECS) (Khan, 2008). Since the same activity or value can have different time intervals which consist of start time and finish time recorded in the event logs, we propose to use fuzzy control to get a crisp time from these time intervals. The general structure of AFTC is shown in Figure 4.5. The time duration and the remaining time are the input side of AFTC. The reason of choosing remaining time as the control to activate the specific fuzzy rule in the AFTC is that the remaining lifetime can be used to get sample measurements for predictions. According to section 3.2.1, each event has a start time and a finish time and each event relates to an activity or a value. Hence, a multi-set of start time and finish time can be derived from event logs. The time duration is calculated by subtraction of finish time and start time in the same case. The remaining time is calculated by subtraction of the finish time of the last case and the finish time of the current case in the same instance. For example in Table 3.1, the time duration of the value "Men" in product X_1 is $00:30 - 00:20 = 10$ minutes, in product X_2 is $02:10 - 02:00 = 10$ minutes. The remaining time of the value "Men" in product X_1 is $01:10 - 00:30 = 40$ minutes, in product X_2 is $02:30 - 02:10 = 20$ minutes.

In Figure 4.5, the inputs of time duration and remaining time are converted into linguistic variables through fuzzifier. The fuzzifier compares the inputs crisp values with certain levels and generates linguistic values of each input variable for inference kernel connected with knowledge base. The knowledge base includes two

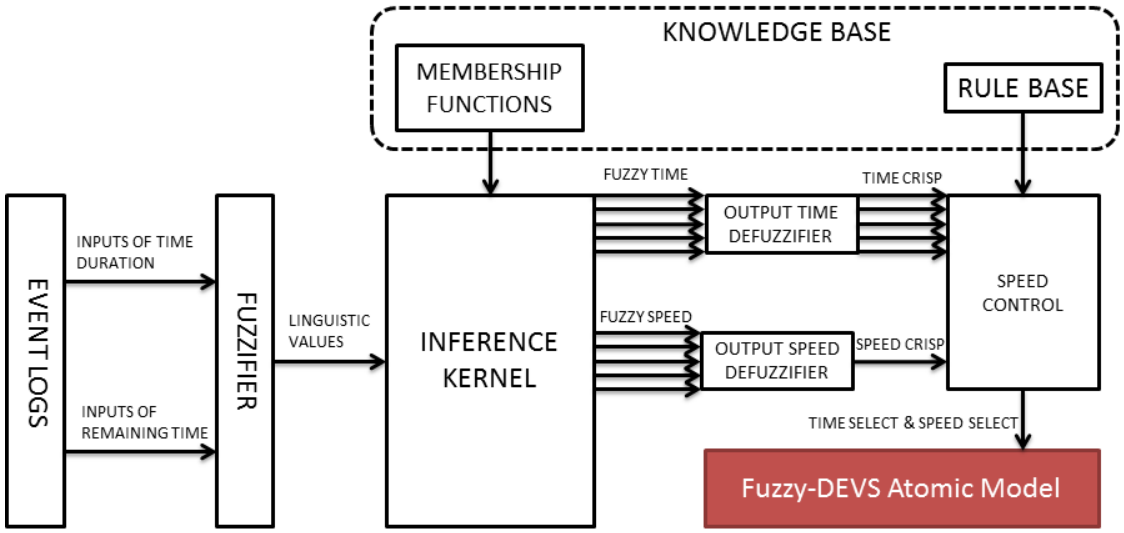


Figure 4.5: The general structure of AFTC.

parts: membership function, which defines the relations between linguistic variables and time variables; rule base, which characterizes the control of remaining time with a set of linguistic control rules. The inference kernel allows human decision to integrate with fuzzy concepts, membership function and inference rules. The defuzzifier converts the fuzzy sets into crisp value. There are two defuzzifiers in this system: time duration and time speed. Five fuzzy time are defuzzified into five crisp time. Five fuzzy speeds are defuzzified into one crisp speed. The defuzzifier is based on the weighted average method. Speed control interprets the crisp speed value and activates one of the crisp time values. Table 4.3 and Table 4.4 show an example of the membership functions for the time duration and remaining time. Table 4.5 presents another example of the rule base to select crisp time. These three tables are designed and proposed to be appropriate in D2FD method.

Table 4.3: Membership functions of input fuzzy time duration

Membership Function - MF	Time Duration
Very Small - VS	0 - 20%
Small - S	20% - 40%
Medium - M	40% - 60%
Big - B	60% - 80%
Very Big - VB	80% - 100%

Based on Table 4.3 and Table 4.4, the set of time is split averagely into five parts. The percentage is calculated based on maximum time (time duration and remaining

Table 4.4: Membership functions of input fuzzy remaining time

Membership Function - MF	Remaining Time
Very Low - VL	0 - 20%
Low - L	20% - 40%
Adequate - A	40% - 60%
High - H	60% - 80%
Very High - VH	80% - 100%

Table 4.5: Illustration of rules applied for time selection

NO	Range of S	Selection of speed	Selection of time T^F
1	0 - 20%	Very Fast	$T1$
2	20% - 40%	Fast	$T2$
3	40% - 60%	Medium	$T3$
4	60% - 80%	Slow	$T4$
5	80% - 100%	Very Slow	$T5$

time) and minimum time (time duration and remaining time). T_{VS}, T_S, T_M, T_B, T_{VB} represent five sets of time duration based on the percentage. T_{VL}, T_L, T_A, T_H, T_{VH} represent five sets of remaining time based on the percentage. The possibility of each set of time is calculated by the frequency of the set and the whole set in equation 4.9.

$$\mu_i = \frac{F_i}{F} \tag{4.9}$$

Where μ is the possibility, F is the frequency of time and i represents the set of time. i can represent VS, S, M, B, VB and VL, L, A, H, VH. Once the possibilities are obtained, the ten sets of time are fuzzed. Moreover, the inference kernel divides the input fuzzy time duration into five groups. Fuzzy time sets are classified as $\{\widetilde{T}_{VS}$, $\widetilde{T}_S$, $\widetilde{T}_M)$, $(\widetilde{T}_{VS}$, $\widetilde{T}_S$, $\widetilde{T}_M$, $\widetilde{T}_B)$, $(\widetilde{T}_{VS}$, $\widetilde{T}_S$, $\widetilde{T}_M$, $\widetilde{T}_B$, $\widetilde{T}_{VB})$, $(\widetilde{T}_S$, $\widetilde{T}_M$, $\widetilde{T}_B$, $\widetilde{T}_{VB})$, $(\widetilde{T}_M$, $\widetilde{T}_B$, $\widetilde{T}_{VB})\}$. Fuzzy speed sets include $\{(\widetilde{T}_{VL}$, $\widetilde{T}_L$, $\widetilde{T}_A$, $\widetilde{T}_H$, $\widetilde{T}_{VH})\}$. The weighted average method is used to transform these fuzzy sets into crisp values. The weighted average method is defined by the following equation:

$$Z = \frac{\sum \mu(\overline{z}) \bullet \overline{z}}{\sum \mu(\overline{z})} \tag{4.10}$$

Where $\overline{z}$ is the mean value in the set of Z and μ represents the possibility. The fuzzy time sets are converted into crisp time values $\{T1, T2, T3, T4, T5\}$. Correspondingly, the fuzzy speed sets are converted into crisp speed value $\{S\}$. Based

on the rule base in Table 4.5, the time T^F can be finally selected as the time of the state in Fuzzy-DEVS atomic model.

In the toy case, imagine that we not only get time duration and remaining time of the value "Men" in product X_1 and X_2, we get a set of time duration {10, 10, 15, 31, 13, 17, 21, 11, 25, 9} and a set of remaining time {30, 10, 35, 25, 19, 12, 38, 28, 41, 25}. The maximum time is 31 and 41 and the minimum time is 9 and 10. Then we get the mean values: $\overline{z}_{VS} = 11.2$, $\overline{z}_S = 15.6$, $\overline{z}_M = 20$, $\overline{z}_B = 24.4$, $\overline{z}_{VB} = 28.8$, $\overline{z}_{VL} = 13.1$, $\overline{z}_L = 19.3$, $\overline{z}_A = 25.5$, $\overline{z}_H = 31.7$, $\overline{z}_{VH} = 37.9$. The possibility can also be obtained: $\mu_{VS} = 0.5$, $\mu_S = 0.2$, $\mu_M = 0.1$, $\mu_B = 0.1$, $\mu_{VB} = 0.1$, $\mu_{VL} = 0.2$, $\mu_L = 0.1$, $\mu_A = 0.3$, $\mu_H = 0.1$, $\mu_{VH} = 0.3$. Based on weighted average method, the crisp time values $T1 = 13.4$, $T2 = 14.6$, $T3 = 16$, $T4 = 17.4$, $T5 = 21.1$, $S = 26.7$. The speed is "Medium" in Table 4.5 so $T^F = T3 = 16$.

4.2.4 Applying Fuzzy Cluster for Fuzzy-DEVS Coupled Model

As the Fuzzy-DEVS atomic model generalizes four functions into fuzzy sets, the Fuzzy-DEVS coupled model also needs to apply fuzzy sets. The problem is the mismatch of the input events and the output events between models or components. Fuzzy cluster, introduced in section 2.4.5, not only forms groups in such a way that data in the same group are similar to each other, but also extends more information which allows for some ambiguity in the data. Here, we propose a new Fuzzy-DEVS coupled model. This model applies fuzzy cluster to generalize three functions into fuzzy sets, i.e. external input coupling, external output coupling, internal coupling. The data can be organized in an n-by-k matrix, where the rows correspond to the objects (or cases) and the columns correspond to the variables. In the data, the variables can be identified as clusters. We propose to use membership coefficients to assign these variables. Here, we do not consider about the human analysis or ontological alignment. Hence, if two variables have the same name or one belongs to one part of another, they can be aligned as a cluster. Moreover, a new select function is proposed appropriate to this new model. This new Fuzzy-DEVS coupled model $\widetilde{N}$ is defined as follows:

$$\widetilde{N} =< X, Y, \widetilde{D}, \widetilde{EIC}, \widetilde{EOC}, \widetilde{IC}, SELECT' > \tag{4.11}$$

Where

- X: input event sets.
- Y: output event sets.

- $\widetilde{D}$: Fuzzy-DEVS component sets.
- $\widetilde{EIC} \subseteq \{((\widetilde{N}, ip_{\widetilde{N}}), (\widetilde{d}, ip_{\widetilde{d}}), (\mu_{EIC}, e)) \mid ip_{\widetilde{N}} \in IPorts, \widetilde{d} \in \widetilde{D}, ip_{\widetilde{d}} \in IPorts_{\widetilde{d}}\}$, fuzzy external input coupling.
- $\widetilde{EOC} \subseteq \{((\widetilde{d}, op_{\widetilde{d}}), (\widetilde{N}, op_{\widetilde{N}}), (\mu_{EOC}, e)) \mid op_{\widetilde{N}} \in OPorts, \widetilde{d} \in \widetilde{D}, op_{\widetilde{d}} \in OPorts_{\widetilde{d}}\}$, fuzzy external output coupling.
- $\widetilde{IC} \subseteq \{((\widetilde{a}, op_{\widetilde{a}}), (\widetilde{b}, ip_{\widetilde{b}}), (\mu_{IC}, e)) \mid \widetilde{a}, \widetilde{b} \in \widetilde{D}, op_{\widetilde{a}} \in OPorts_{\widetilde{a}}, ip_{\widetilde{b}} \in IPorts_{\widetilde{b}}\}$, fuzzy internal coupling.
- $SELECT' : (\widetilde{d}, max(M_{\widetilde{N}})) \mid \widetilde{d} \in \widetilde{D}, \mu_{EIC}, \mu_{EOC}, \mu_{IC} \in M_{\widetilde{N}}$.

In this new Fuzzy-DEVS coupled model $\widetilde{N}$, Fuzzy-DEVS component sets $\widetilde{D}$ represent Fuzzy-DEVS atomic models in the equation 2.4. X and Y are the sets of input events and output events from all these components. Three functions EIC, EOC, IC are converted into fuzzy sets. In the $\widetilde{EIC}$, the coupled model $\widetilde{N}$ has a corresponding input port $ip_{\widetilde{N}}$. The component $\widetilde{d}$ has a corresponding input port $ip_{\widetilde{d}}$. The coupling between these two ports has a membership coefficient μ_{EIC} (between 0 and 1) with the elapsed time e. In the $\widetilde{EOC}$, the coupled model $\widetilde{N}$ has a corresponding output port $op_{\widetilde{N}}$. The component $\widetilde{d}$ has a corresponding output port $op_{\widetilde{d}}$. The coupling between these two ports has a membership coefficient μ_{EOC} (between 0 and 1) with the elapsed time e. In the $\widetilde{IC}$, $op_{\widetilde{a}}$ represents the output port of the component $\widetilde{a}$ and $ip_{\widetilde{b}}$ represents the input port of the component $\widetilde{b}$. Between these two ports, there is a membership coefficient μ_{IC} (between 0 and 1) with the elapsed time e. All the three membership coefficients μ_{EIC}, μ_{EOC}, μ_{IC} are calculated by the Dependency Method. The membership coefficient comes from the possibility of the corresponding output function. Meanwhile, these membership coefficients are also related to the state and the time. The similar activities or values perhaps happen in different models or components. Hence, the corresponding transitions and output functions probably have different possibilities. Therefore, the membership coefficients may change with the change of the elapsed time e. The original function $SELECT$ is no more suitable for this new Fuzzy-DEVS coupled model. The new $SELECT'$ proposes to make a selection of the component $\widetilde{d}$ which has the maximum membership coefficient. If there are multiple input events triggered, the maximum membership coefficient refers to μ_{EIC} and μ_{IC}. If there are multiple output events triggered, the maximum membership coefficient refers to μ_{EOC}.

In the toy case, we can finally construct a Fuzzy-DEVS coupled model depicted in Figure 4.6. This coupled model is represented by a big box with two ports $IPort$ and

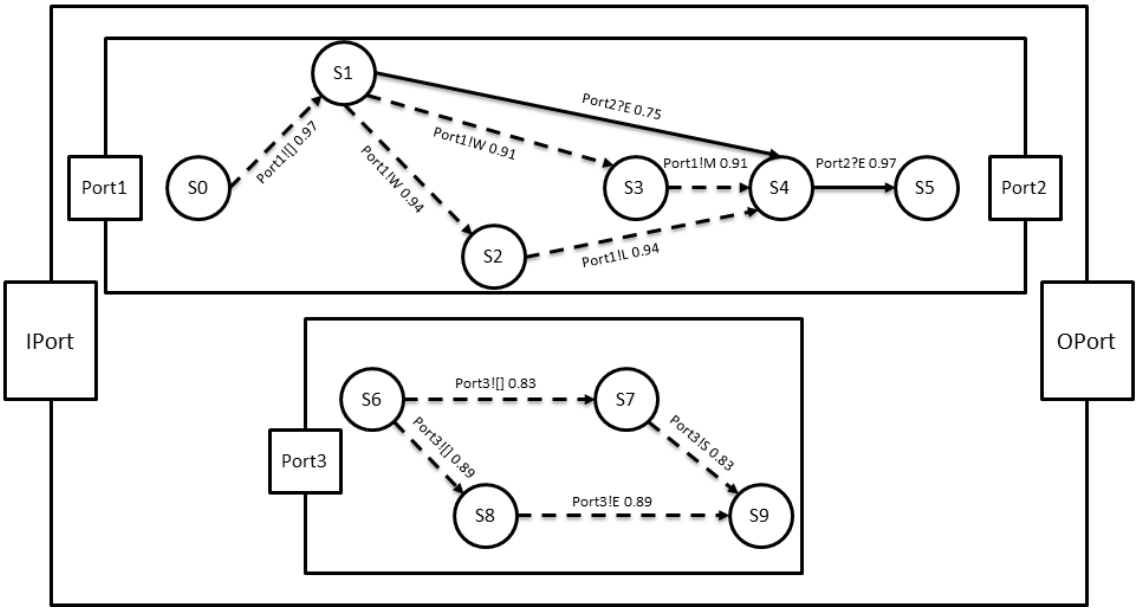


Figure 4.6: Fuzzy-DEVS coupled model in the toy case.

$OPort$. It consists of two Fuzzy-DEVS atomic model as same as the ones in Figure 4.4. In the upper atomic model, the input events have $Port2?E$, the output events include $Port1![\,]$, $Port1!W$, $Port1!M$, $Port1!L$. In the atomic model below, there is no input events and the output events include $Port3!S$, $Port3!E$. When considering about $\widetilde{IC}$ between two components, the situations can be distinguished by the state and time. In order for explanation, we assume that the Fuzzy-DEVS atomic model in Figure 4.7 is inside the Fuzzy-DEVS coupled model in Figure 4.6. For the state, if the state is $S7$, the membership coefficient of $Port2?E$ and $Port3!E$ is 0. If the state is $S8$, the membership coefficient of $Port2?E$ and $Port3!E$ is 0.89. It means that the customer who likes electric prefers TV&Video. For the time, if the elapsed time reaches ta_{S11} and the state is $S11$, the membership coefficient of $Port2?E$ and $Port4!E$ is 0.92. When the elapsed time reaches to $ta_{S11} + ta_{S12} + ta_{S13}$ and the state goes to $S13$, the membership coefficient of $Port2?E$ and $Port4!E$ is 0.94. As a result, μ_{IC} changes from 0.92 to 0.94. When considering about $\widetilde{EOC}$, the membership coefficients are related to all the output events for example the membership coefficient of $OPort!M$ is 0.91, $OPort!L$ is 0.94, $OPort!S$ is 0.83. Since there is only one Fuzzy-DEVS coupled model, $\widetilde{IC}$ is related to $\widetilde{EOC}$ in the other coupled models. $SELECT'$ is triggered when the output functions occur from the atomic model below in Figure 4.6 and in Figure 4.7. If the state is $S8$ and $S11$ at the same time, $Port2?E$ selects $Port4!E$ to be connected with each other instead of $Port3!E$ because the membership coefficient 0.92 is the maximum.

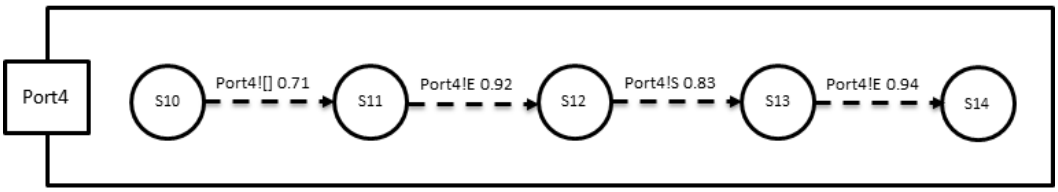


Figure 4.7: The hypothetical Fuzzy-DEVS atomic model in Figure 4.6.

4.3 Conclusion

This section presents the important part of the D2FD method. The approach from TS to Fuzzy-DEVS consists of several methods. An improved region-based approach is proposed to discover the regions in TS associated with a c-groups method. The c-groups method is able to improve the performance of this approach. Several rules are proposed to transform the elements of event logs and TS into Fuzzy-DEVS. Dependency method is proposed to convert three functions of DEVS into fuzzy sets. Possibility Measure defuzzifies these fuzzy sets into crisp values for simulation. AFTC is proposed to first fuzzify the time duration and the remaining time and then defuzzify into crisp time value. A new formalism of Fuzzy-DEVS coupled model is proposed which extends the original Fuzzy-DEVS formalism. Fuzzy cluster is applied to connect the components to form the entire Fuzzy-DEVS coupled model. In the toy case, we finally construct a Fuzzy-DEVS coupled model depicted in Figure 4.6. The simulation results of this model are able to realize the goals in section 3.3.1. After explain the whole methodology of D2FD method, we continue to present the implementation of this method on the computer in chapter 5.

Chapter 5

Implementation of D2FD Method

5.1 Introduction

Chapter 3 and chapter 4 have described the D2FD method from event data to Fuzzy-DEVS model. This method is implemented as a plugin in the process mining framework (ProM). According to Figure 1.2, the simulation tool SimStudio is used for the simulation of the Fuzzy-DEVS model. In order to evaluate the implementation of D2FD method, two real case studies are provided coming from Dutch Employee Insurance Agency and Rabobank Group ICT (Information and Communication Technologies) respectively. This chapter is organized as follows. Section 5.2 introduces the development environments ProM and SimStudio. Section 5.3 presents the application of the whole process from event data to Fuzzy-DEVS models and its simulation. The plugin is proposed based on D2FD method and implemented on ProM. The case study of Dutch Employee Insurance Agency is used to support the explanation of the application. The case study of Rabobank Group ICT gives some other modeling and simulation results. Section 5.4 concludes this chapter.

5.2 Development Environment

D2FD method is implemented based on the plugin of ProM. On ProM, there are a lot of plugins which can provide different kinds of process mining techniques. Besides, there are also a big amount of process mining tools. The simulation of D2FD method is based on the integration of SimStudio. There are also a lot of DEVS simulators currently available for different kinds of DEVS extensions. In this section, we present the other process mining tools and DEVS simulators to make a comparison.

5.2.1 Process Mining Framework (ProM)

Most of the existing Business Intelligence (BI) software products are data-centric and limited by rather simple forms of analysis. Some other process mining tools are developed based on the expert assumption and special enterprise requirements. For this reason, ProM (Van der Aalst, 2011) is proposed as a "plug-able" environment for process mining using XES as input format. It provides many kinds of analysis and supports many different types of models. The basic idea of ProM is to provide a basic framework to allow for all kinds of process mining techniques. When people develop new process discovery algorithms, they do not need to worry about extracting, converting and loading event data. This event data is based on XES standard as explained in section 2.3.1. Moreover, ProM can also provide the standards of model types for example TS and Petri Net. ProM is extremely powerful and it develops new functionality every day.

Table 5.1: Some of the process mining plug-ins in ProM 6.7

Plug-in	Description
Alpha miner	Discovers a Petri net using the α-algorithm, see Section 2.3.3.1
Heuristic miner	Discovers a C-net using heuristic mining, see Section 2.3.3.3
Transition system miner	Discovers a transition system based on a state representation function and a log, see Section 3.4
Transition system to Petri net	Uses state-based regions to create a Petri net based on a transition system, see Section 2.3.3.2
Convert CSV to XES	Transforms the CSV file into the event logs, see Section 5.3.2

The current version of ProM is 6.7[1]. It is implemented in Java and can be downloaded free of charge. ProM 6.7 is distributed as a package. This package consists of four parts which are framework, plugins, contexts and models. The framework is based on the GNU Public License (GPL) open source license. The plugins are distributed as separate packages using the Lesser GNU Public License (L-GPL) open source license. This means that the software which uses the core needs to follow the GPL license. However, if a plug-in is built based on a changed version, it is required to distribute this changed plug-in under the L-GPL license as well. The contexts are used for collecting process mining algorithms. The models are used for

[1]http://www.promtools.org/doku.php

its construction and visualization. On ProM 6, hundreds of plugins are implemented for the techniques of process mining. Table 5.1 presents some of the plugins which are explained in this thesis. The algorithms of these plugins have been explained in this thesis. Later in section 5.3, the interface of ProM 6 will be shown together with the implementation of D2FD method.

5.2.2 Other Process Mining Tools

ProM provides a lot of plugins which develop different kinds of process mining techniques. Besides, there are also several research groups developing process discovery tools. Table 5.2 presents some of the process mining tools currently available (Van der Aalst, 2011).

Table 5.2: Examples of process mining tools

Product name	Organization
ARIS Process Performance Manager	Software AG (www.softwareag.com)
Enterprise Visualization Suite	Businesscape (www.businesscape.no)
Disco	Fluxicon (www.fluxicon.com)
Interstage BPME	Fujitsu (www.fujitsu.com)
OKT Process Mining suite	Exeura s.r.l. (www.exeura.com)
ProcessAnalyzer	QPR (www.qpr.com)
Reflect—one	Pallas Athena (www.pallas-athena.com)
Reflect	Futura Process Intelligence (www.futuratech.nl)

The ARIS Process Performance Manager by Software AG can provide some process mining techniques and it focuses mainly on performance analysis (drilling down to instance level, bench marking and dashboards) (Scheps, 2011). The Enterprise Visualization Suite by Businesscape puts emphasis on the analysis of SAP (Systems Applications and Products) supported business processes. Disco by Fluxicon is a stand-alone process mining tool. The fuzzy mining approach is applied in this tool with a high performance (Günther and Van der Aalst, 2007). Interstage BPME (Business Process Management through Evidence) by Fujitsu provides a service for Interstage Automated Process Discovery. The OKT Process Mining suite by Exeura uses the process discovery approach based on the clustering of log traces (Greco et al., 2006). QPR ProcessAnalyzer uses a process discovery algorithm inspired by the α-algorithm and heuristic mining. Several Finish hospitals have already applied this tool. Reflect—one by Pallas Athena and Reflect by Futura Process Intelligence are

essentially the same product. They are used as mature and stand-alone tools for process mining. The genetic mining approach and the filtering of infrequent behavior are used as two different discover algorithm. Moreover, Reflect can create social networks based on handover of work.

5.2.3 Simulation Engine SimStudio

The SimStudio (Traoré, 2008) is used as a simulator, implemented in Java programming language, that manages the communications between models and manage time. The specifications of models in SimStudio is based on either the Classic DEVS or the Parallel DEVS formalisms. Figure 5.1 presents a DEVS meta-model using UML diagram in SimStudio that offers a Model abstract class, from which derive an Atomic Model abstract class and a Coupled Model abstract class and a Port abstract class which is extended by Input and Output. The models of user must derive from these sub-classes and override the abstract methods. While the Port abstract class is composed of Model abstract class, the coupled model aggregates the Model. In the Model class, there are attributes of *sim* (simulator that drives the model), *name* (name of the model), *id* (identify of the model), X (list of inputs) and Y (list of outputs). The operations are based on adding, setting and getting of these attributes. In the Atomic Model sub-class, there is *state* attribute (current state of the model). The operations of $addStateVariable$, $getVar$ and $setVar$ are used for the adding, setting and getting of the state of the model. *lambda* generates outputs, *deltaInt* determines the next state of the model when an internal transition occurs, *deltaExt* determines the next state of the model when an external transition occurs, *ta* determines the duration of the current state. In the Coupled Model sub-class, there are the attributes of $subModels$ (list of sub-models), EIC (list of external input coupling), EOC (list of external output coupling) and IC (list of internal coupling). The operations include the adding of these attributes. Besides, *getLinkedOutput* gets the linked outputs, *getLinkedInput* gets the linked inputs, *getLinkedInernalPort* gets the internal linked port for a given port, *getLinkedPort* gets all the linked ports for a given port. In the Port class, the attributes are *model*, *value* and *name*. The operations focus on setting and getting these attributes.

The models in Figure 5.1 are then executed by predefined engines. In order to communicate between models, the Message abstract class defines some specifications of the various types of messages. The types of messages consist of $I\_Messages$ (the initialization message), $S\_Message$ (the internal transition message), $X\_Message$ (the external transition message) and $Y\_Message$ (the output message). Figure 5.2

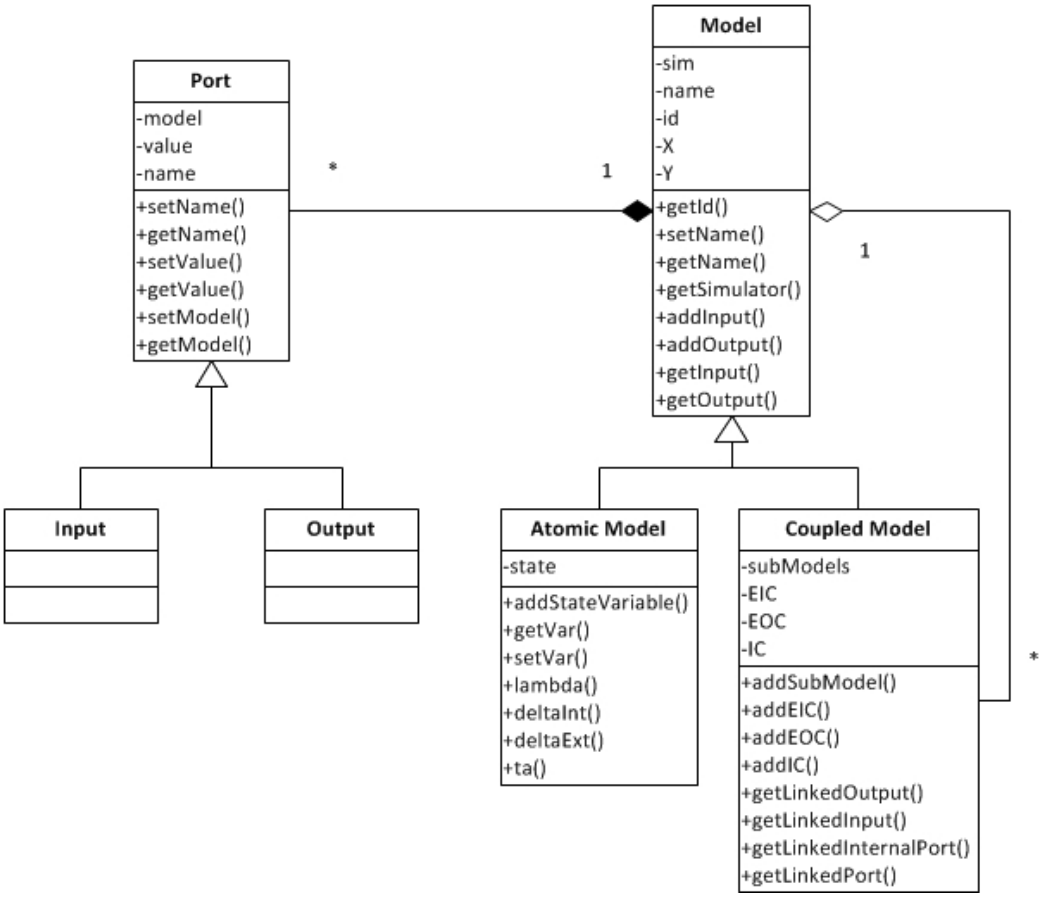


Figure 5.1: The UML diagram of Model and Port in SimStudio.

presents the UML diagram of Simulator Engines in SimStudio. The Simulator class drives an atomic models. *model* is its attribute. *init* initializes the simulation, $getModel$ returns the driven atomic model, $internalTransition$ handles an internal transition and $externalTransition$ handles an external transition. The Coordinator class drives a coupled model and it is a specific simulator. There are two attributes: *model* represents the driven coupled model and *subjects* represents list of the sons (simulators). $getModel$ returns the model of the coordinator, *addSubject* adds a son to the simulator, *init* initializes the coordinator and sends messages to all the sons, $internalTransition$ handles an internal transition, $externalTransition$ sends the messages to EIC, $transfert$ sends the messages to EOC and IC and $updateTn$ calculates the date of the next event. The Simulator and Coordinator are inherited from the AbstractSimulator class which contains the basis of them. In the Abstract-Simulator class, there are the attributes of tn (date of next event), tl (date of last event), e (elapsed time since last event) and *parent* (parent simulator). The operations are based on handling of these attributes. Especially, $handleMessage$ handles

the incoming four types of messages. The RootCoordinator class is identified as the central manager to trigger the simulation. The attribute *sim* represents the triggered simulator. *init* initializes the simulator and *run* launches the simulation. Moreover, the DEVS_type abstract class defines the types of input and output. The types include integer, character, interval of real numbers or integers, string, real number and enumeration of objects.

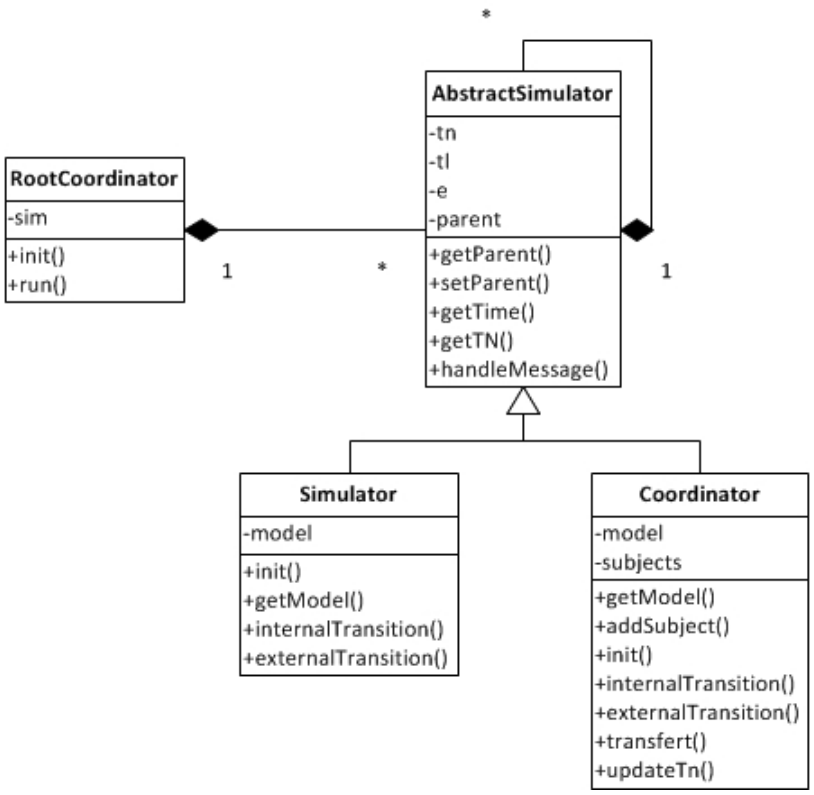


Figure 5.2: The UML diagram of Simulator Engines in SimStudio.

5.2.4 Other DEVS Simulators

DEVS simulator is able to improve the performance of DEVS modeling and simulation for the model users. Here we give some other DEVS simulators. Each simulator has a certain functionality to make it more efficient in the certain problem areas. These simulators all conforms to the DEVS specification.

ADEVS[2] is a simulator which focuses on performance and lightweightness. The code language is based on C++. The simulator is specialized to the models due to

[2]https://web.ornl.gov/∼nutarojj/adevs/

the extensive use of templates. This tool has been developed by the group of Zeigler in the University of Arizona.

CD++[3] is written in C++ and is focused on the Cell-DEVS formalism (Wainer and Giambiasi, 2001). This tool is built as a hierarchy of models, each of them related to a simulation entity. This tool also allows for some graph-based notations. In addition, Jacques and Wainer (Jacques and Wainer, 2002) propose an approach of mapping of the Petri Net modeling formalism into the DEVS modeling formalism using CD++.

DEVS-Suite[4] (Kim et al., 2009) is developed by the language JAVA. This tool contains an additional simulation viewer and a tracking environment. It shows a high portability for both the model and the simulator.

MS4 ME (Seo et al., 2013) is written in Java on the platform Eclipse. This tool is not open source and free. However, several important functions are not available such as hierarchical coupled models.

Python DEVS (Van Tendeloo and Vangheluwe, 2014) is a simulator written in Python. This tool has a small code base but it is extremely lightweight. It has been developed by Modeling, Simulation and Design Lab (MSDL) in the University of McGill.

VLE[5] (Virtual Laboratory Environment) (Quesnel et al., 2009) is used a complete and powerful simulator to analysis complex systems. This tool connects the heterogeneous models and simulates them based on a formal basis. It also defines experimental schemes for loading simulation data and saving models in XML. The provided libraries allow the people to develop personal programs. This tool is developed by the computer science lab in the University of the Littoral Opal Coast.

XSY[6]is written in Python. The main feature is the verification engine. This tool uses a scanning list as scheduler. It is developed by Moon Ho Hwang and current version is 1.0.0.

Van Tendeloo (Van Tendeloo, 2013) has made a test of performance for all the above simulators except SimStudio. The test is based on several criteria to compare the efficiency and functionality of these simulators. We apply the same test of traffic model on SimStudio and get some results of performance. Table 5.3 and Table 5.4 have shown the general results of the comparison. IDE represents Integrated Development Environment. GUI represents Graphical User Interface. CLI represents Commend

[3]http://cell-devs.sce.carleton.ca/mediawiki/index.php/Main_Page
[4]http://devs-suitesim.sourceforge.net/
[5]http://www.vle-project.org/
[6]https://code.google.com/archive/p/x-s-y/

Line Interface. The test of traffic model on SimStudio takes 0.03 seconds which is close to ADEVS so the performance of SimStudio is fast. As a result, as ProM provides general interface for kinds of model, SimStudio is easy to be planted in the plugin on ProM and be adapted to Fuzzy-DEVS.

Table 5.3: General comparison of first part of DEVS simulator

	ADEVS	CD++	DEVS-Suite	MS4 ME
Formalism	DynDEVS	Cell-DEVS	PDEVS	PDEVS
IDE	no	optional	no	yes
GUI	no	optional	yes	yes
Parallel	yes	yes	no	no
Distributed	no	yes	no	no
Stop condition	time	time	steps	tiem/steps
Performance	fast	medium	slow	medium
Interactivity	no	file	GUI	GUI
Debugging	medium	medium	medium	medium
Enforcement	no	no	no	partial

Table 5.4: General comparison of second part of DEVS simulator

	PyDEVS	VLE	XSY	SimStudio
Formalism	CDEVS	DSDE	CDEVS	PDEVS
IDE	no	optional	no	optional
GUI	no	optional	no	no
Parallel	no	no	no	no
Distributed	no	yes	no	yes
Stop condition	function	time	time	time
Performance	slow	fast	slow	fast
Interactivity	no	no	CLI	no
Debugging	easy	easy	medium	easy
Enforcement	no	yes	no	no

5.3 Application of the D2FD Method

As explained, ProM is used as the platform to implement D2FD method. A new plugin called *Convert to $Fuzzy - DEVS$ using Regions* is designed on ProM. This plugin[7] is synchronized on the server managed by the Architecture of Information

[7]https://svn.win.tue.nl/repos/prom/Packages/TS2DEVS

Systems (AIS) group at Eindhoven University of Technology (Tu/e). SimStudio as the simulation engine is integrated with this plugin. The main function of this plugin is to convert TS into Fuzzy-DEVS model and simulate this model. Moreover, plugin *Convert CSV to XES* and plugin *Transition system miner* in Table 5.1 are used to transform from CSV to event logs and from event logs to TS respectively. According to D2FD method explained in Chapter 3 and 4, we use a real case study of Dutch Employee Insurance Agency to explain how to get Fuzzy-DEVS model with simulation results from CSV files.

5.3.1 Case Study of Dutch Employee Insurance Agency

The case study is conducted on real event data collected from an employee insurance agency in Netherlands. This agency is the merger of two former organizations that were responsible for respectively financial support of unemployed and the brokers function on the labor market. Now these two functions are combined. The unemployed gets help of finding a new job and UWV also checks if sufficient efforts are made to get another job. From the description[8], the agency is interested in insight and recommendations of event data and two main goals can be captured:

1. How the channels are being used?
2. When are customers moving from one contact channel to the next?

Five CSV files are given to measure different aspects of customers behavior:

- *Question.csv* records the behavior of customers when asking questions.
- $Werkmap - message.csv$ records the behavior of customers sending messages through a digital channel.
- $Clicks\_Logged\_In.csv$ records the behavior of known customers when using the website.
- $Clicks\_NOT\ Logged\_In.csv$ records the behavior of unknown customers when using the website.
- *Complaints.csv* records the behavior of customers when complaining.

These files are also defined by the following nine dimensions (Jalali, 2016):

[8]https://data.4tu.nl/repository/collection:event_logs_real

- *Date*: to capture the calendar information. This dimension has a hierarchy of the following levels: Year → Half Year → Quarter → Month → Week → Day..
- *ComplainType*: to capture the complain information. This dimension has a hierarchy of the following levels: Complain 1 → Theme → Subtheme → Topic.
- *Question*: to capture the question information. This dimension has a hierarchy of the following levels: Question 1 → Theme → Subtheme → Topic.
- *Page*: to capture the visited page information. This dimension has a hierarchy of the following levels: WebSite 1 → vhost (host address) → (Page) Name.
- *Channel*: to capture the available channels for communication.
- *Customer*: to capture customer information.
- *Session*: to capture the session information.
- IP: to capture the IP information.
- $Message$: to capture the Message information.

Heidari and Assy (Heidari and Assy, 2016) take these data for process mining analysis. They propose a new methodology with three major phases before analysis and present the analysis of the result on the click data. Most of studies make a complete analysis for data but the underlying relationships between data are not identified. In another words, the modularity is not taken into account.

For this reason, we propose to use D2FD method with three plugins on ProM to achieve these two goals. Two CSV files *Question.csv* and $Werkmap - message.csv$ are selected. These two files can be opened by Microsoft Excel. Figure 5.3 shows part of event data in *Question.csv*. There are 123,404 events. Each event has a reference of $CustomerID$ and 16 attributes. Figure 5.4 shows part of event data in $Werkmap - message.csv$. There are 66059 events. Each event has a reference of $CustomerID$ and 7 attributes.

Customer	AgeCa	Ge	ContactDate	ContactT	ContactT	Question	QuestionSubt	QuestionTopic_EN
1876596	18-29	M	24/08/2015	14:00:34.	14:05:23.	WN WW	Application (W	Job Duty: Should I apply if I go back to work?
722160	18-29	M	27/08/2015	11:50:08.	11:55:34.	WN WW	Payment	When is/are transferred my unemployment benefits?
1536943	18-29	M	20/08/2015	11:01:16.	11:09:11.	WN WW	Payment (WW	When is/are transferred my unemployment benefits?
1816718	18-29	M	27/08/2015	12:19:28.	12:25:51.	WN WW	Payment (WW	When is/are transferred my unemployment benefits?

Figure 5.3: The screen-shot of *Question.csv*.

CustomerID	AgeCategory	Gender	Office_U	Office_W	EventDateTim	EventType	HandlingChannelID
2032131	18-29	M	271	271	2015-11-02 0:	Werkmap me:	1
2032131	18-29	M	271	271	2015-11-05 2:	Werkmap me:	1
2032131	18-29	M	271	271	2015-11-06 1:	Werkmap me:	1
2085395	18-29	V	280	280	2015-10-20 2:	Werkmap me:	2

Figure 5.4: The screen-shot of $Werkmap - message.csv$.

5.3.2 Convert CSV to XES

According to guideline 15 and 16 in section 3.2.2, both of two event data have the start time and the finish time. The instance is identified by $CustomerID$. The events are first ordered by instance and then ordered by start time. From $Question.csv$, we construct the conceptual structure shown in Figure 5.5. All the words in the figures are translated from Dutch language to English language. Here, we only give the useful part of the structure. WN represents the Netherlands agency and WW represents the human resource. $Internet\ Helpdesk$ and WW belong to the property of $QuestionTheme$. Each one has several values on the property of $QuestionSubtheme$.

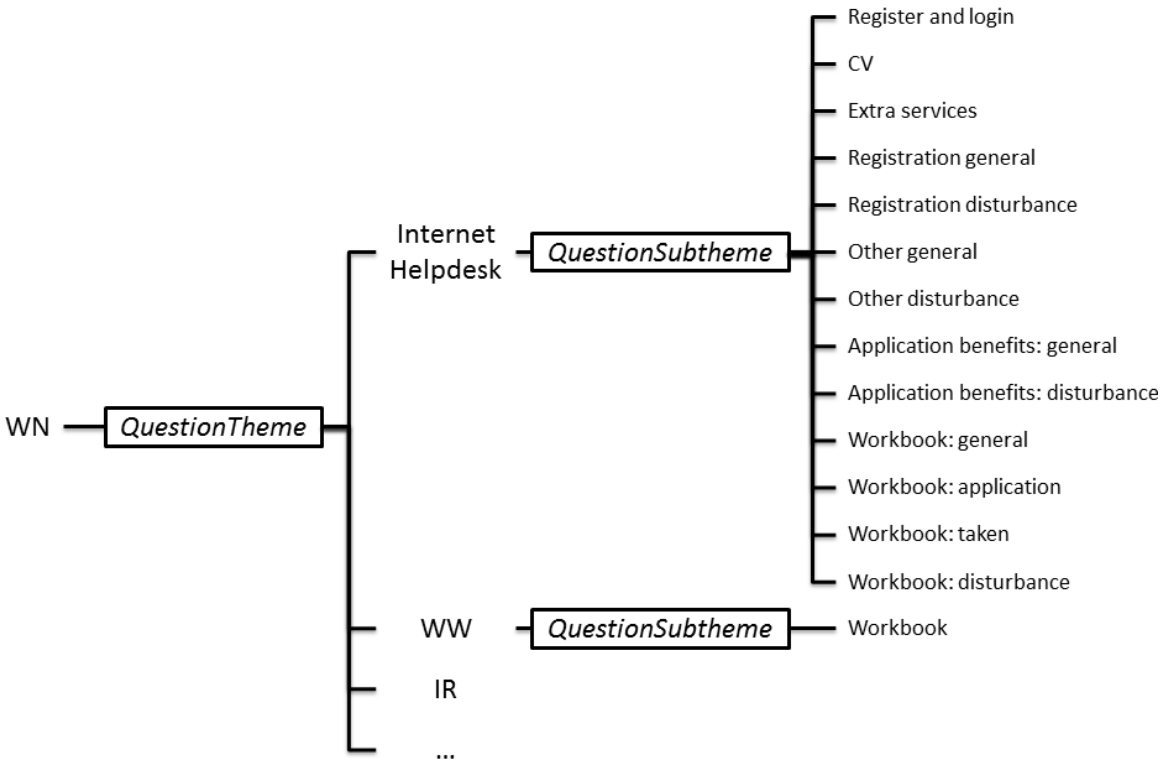


Figure 5.5: The conceptual structure of $Question.csv$.

From $Werkmap - message.csv$, we construct the conceptual structure shown

in Figure 5.6. There is only one *Internet Helpdesk* with one *EventType* property of *Workbook* : *message*. *Workbook* : *message* has a variable *Channel* between 1 and 2. *Channel* is the property and 1 and 2 are the values. When we look at these two conceptual structures, *Workbook* : *message* has a stronger relationship with *Workbook* : *general*, *Workbook* : *application*, *Workbook* : *taken* and *Workbook* : *disturbance*. Hence, the child of *Workbook* : *message* which is the variable *Channel* is identified as private value. All the other values are identified as public values.

Figure 5.6: The conceptual structure of $Werkmap - message.csv$.

In order to select process instance, the interesting level of two conceptual structures are *QuestionSubtheme* property and *Channel* property. As a result, the final selected public values are: *Registration general*, *Registration disturbance*, *Other general*, *Other disturbance*, *Request distribution* : *general*, *Request distribution* : *disturbance*, *Workbook* : *general*, *Workbook* : *application*, *Workbook* : *taken*, *Workbook* : *disturbance*, *Workbook*. *Registerandlogin*, *CV* and *Extra services* are not interesting so they are not selected. The final selected private values are: *Channel* 1, *Channel* 2.

The plugin *Convert CSV to XES*[9] on ProM is used to transform event data to event logs. This plugin is designed by Mannhardt F., Tax N. and Schunselaar D.M.M. from Tu/e. Figure 5.7 presents the initial screen of ProM 6 if ProM is started. From this screen, we can load logs and models through the button "import". The *Question.csv* and $Werkmap - message.csv$ are loaded in this screen. If an event log is loaded, we can also view the underlying data in different ways. Another operation is to apply a plugin and start doing all kinds of analysis. By pressing the triangle button, we go to the plugin screen as shown in Figure 5.8. All the available plugins are shown in the middle chart and *Convert CSV to XES* is shown

[9]https://svn.win.tue.nl/repos/prom/Packages/CSVImporter/

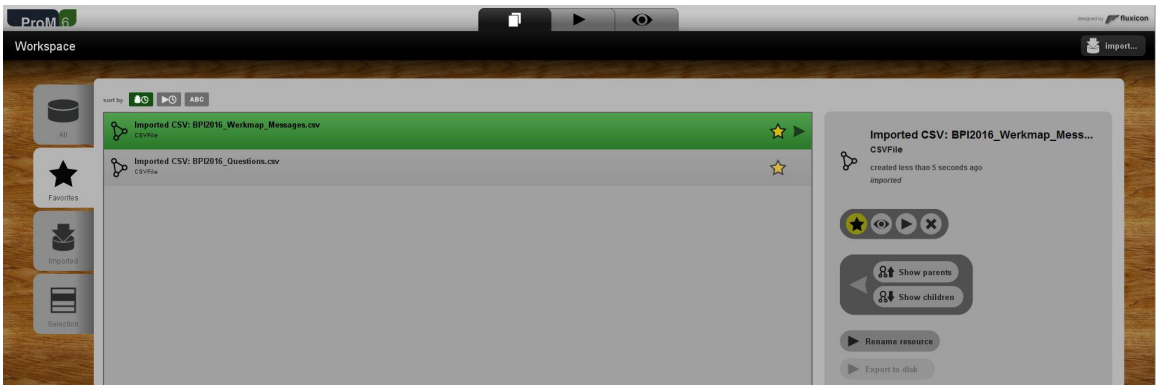


Figure 5.7: The initial screen of ProM 6.

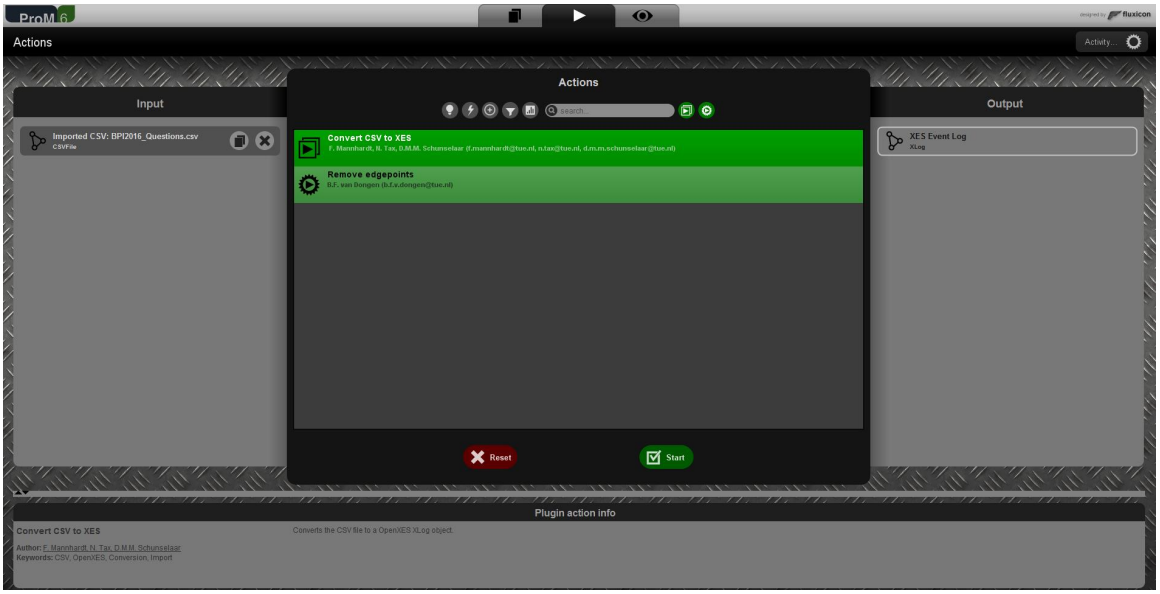
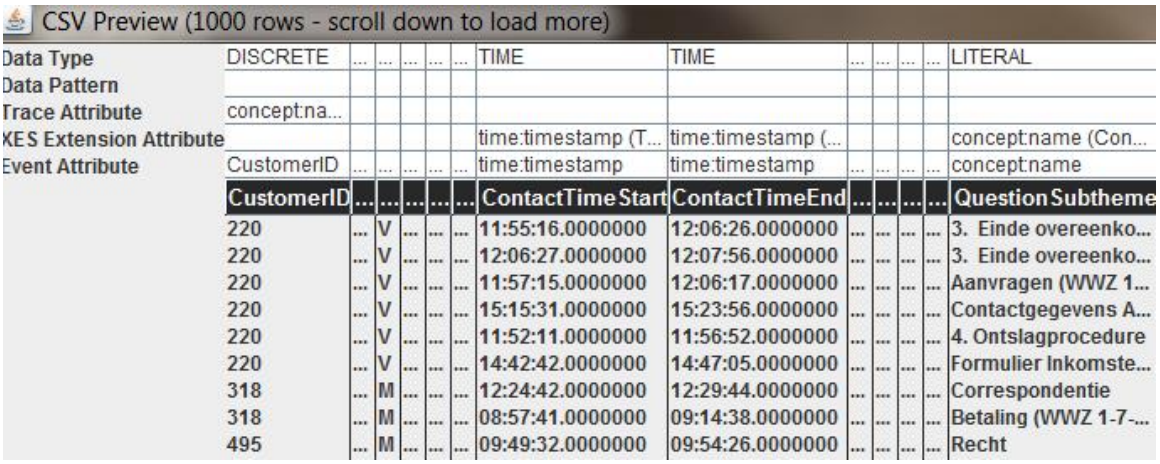


Figure 5.8: The plugin screen of *Convert CSV to XES* on ProM 6.

in the screen. The left side is the input where we put *Question.csv*. The right side is the output where we get the corresponding event logs. The bottom side describes the selected plugin. Once we press the button "play", *Question.csv* is transformed through the screen as shown in Figure 5.9. *Customer ID* is converted to trace in event logs. *ContactTimeStart* is converted to the extension of start timestamp and *ContactTimeEnd* is converted to the extension of end timestamp. The interesting level is found in the *QuestionSubtheme* property and it is transformed into the extension of concept name in event logs. Same for $Werkmap-message.csv$, *Customer ID* is converted to trace in event logs. *EventTimeStart* is converted to the extension of start timestamp and *EventTimeEnd* is converted to the extension of end timestamp. *Eventtype* property is transformed into the extension of concept name in


CSV Preview (1000 rows - scroll down to load more)

Data Type	DISCRETE					TIME	TIME					LITERAL
Data Pattern												
Trace Attribute	concept:na...											
XES Extension Attribute						time:timestamp (T...	time:timestamp (...					concept:name (Con...
Event Attribute	CustomerID					time:timestamp	time:timestamp					concept:name
	CustomerID					**ContactTimeStart**	**ContactTimeEnd**					**QuestionSubtheme**
	220		V			11:55:16.0000000	12:06:26.0000000					3. Einde overeenko...
	220		V			12:06:27.0000000	12:07:56.0000000					3. Einde overeenko...
	220		V			11:57:15.0000000	12:06:17.0000000					Aanvragen (WWZ 1...
	220		V			15:15:31.0000000	15:23:56.0000000					Contactgegevens A...
	220		V			11:52:11.0000000	11:56:52.0000000					4. Ontslagprocedure
	220		V			14:42:42.0000000	14:47:05.0000000					Formulier Inkomste...
	318		M			12:24:42.0000000	12:29:44.0000000					Correspondentie
	318		M			08:57:41.0000000	09:14:38.0000000					Betaling (WWZ 1-7-...
	495		M			09:49:32.0000000	09:54:26.0000000					Recht

Figure 5.9: The screen of the plugin *Convert CSV to XES*.

event logs. At last, we get two event logs *Question.xes* and $Werkmap-message.xes$.

5.3.3 Mine Transition System

Two event logs *Question.xes* and $Werkmap - message.xes$ are obtained from *Question.csv* and $Werkmap - message.csv$. In order to get TS models from event logs, the plugin *Transition system miner*[10] designed by Verbeek H.M.W. at Tu/e is chosen. Figure 5.10 presents the screen of choosing this plugin. The event log from *Question.xes* is added in the input side. The plugin is chosen in the middle. There are four output objects: Mined Transition System represents the generated TS model; Weights represent the weight of the transition in TS calculated by the frequency of events; Start states represent the initial start in TS; Accept states represent the final state in TS. The description of this plugin is on the bottom side. In the configuration of this plugin, we choose the set abstraction so each set represents one activity or one value. In TS, there is a "state explosion" problem. A simple process which has 10 parallel activities can construct a TS model with $2^{10} = 1024$ states and $10 \times 2^{101} = 5120$ transitions. For this reason, we set the collection size limit to 1 in order to get the minimum number of the states.

Once we press the "start" button, a TS model is generated on ProM 6. Figure 5.11 presents the TS model from *Question.csv*. The initial state is at the top of the model depicted by the dotted circle. The other states represent the 11 selected public values depicted by the solid circle. The arcs connecting the circles represent the transitions. The thickness of the arcs represents the weight of the transition.

[10]https://svn.win.tue.nl/repos/prom/Packages/StreamTransitionSystemsMiner/

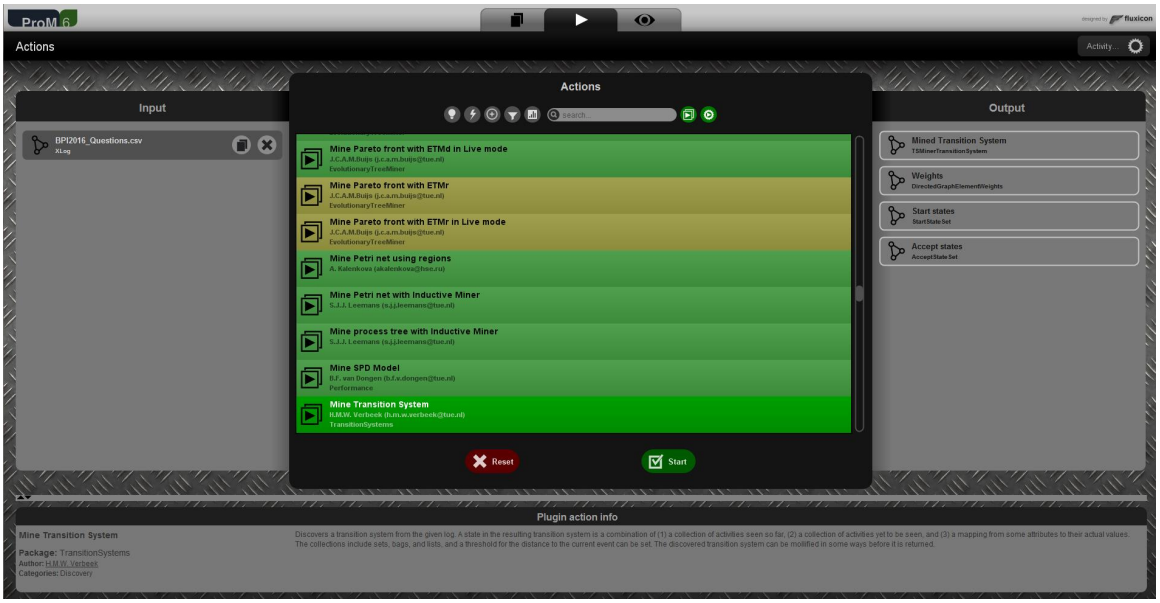


Figure 5.10: The screen of the plugin *Transition system miner*.

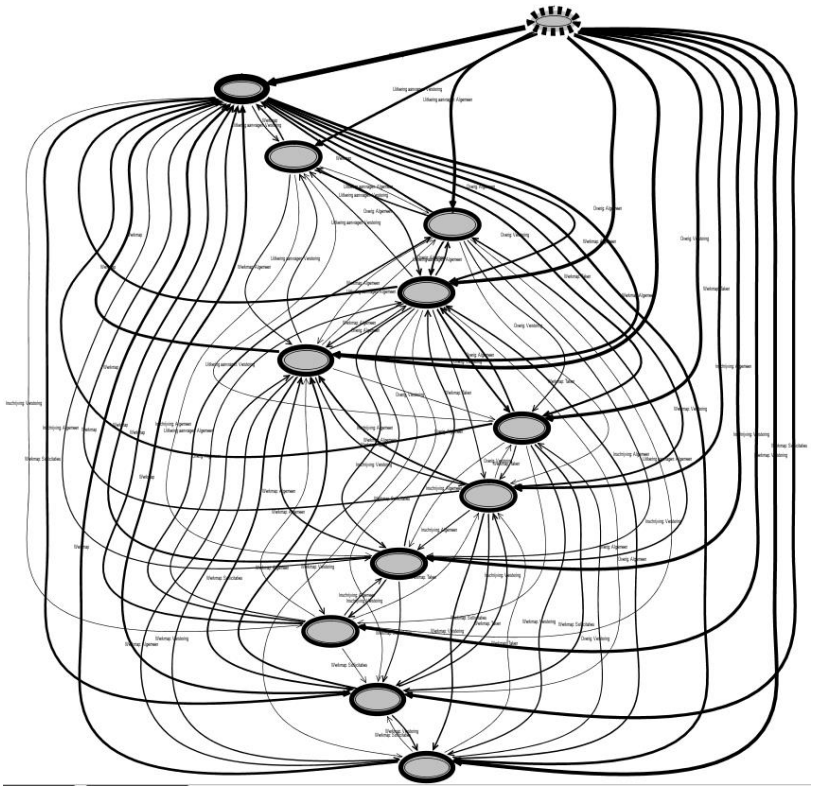


Figure 5.11: The TS model from *Question.csv* on ProM 6.

By using the same procedure, the TS model from $Werkmap - message.csv$ can be mined. Here, we do not describe two TS models in detail. Later in the Fuzzy-DEVS

atomic model, a more detailed explanation is shown.

5.3.4 Convert to Fuzzy-DEVS From TS

Followed by the two TS models, we apply the proposed plugin *Convert to Fuzzy−DEVS using Regions*[7]. The interface of this plugin is presented in Figure 5.12. In the input side, the event logs and TS model from *Question.csv* are added. The reason to put the event logs is to extract more information including time and frequency, which TS model does not have. In the middle, we select the proposed plugin. In output side, Fuzzy-DEVS represents the model and state represents the label of the state. In the bottom, the author and its contact information are given.

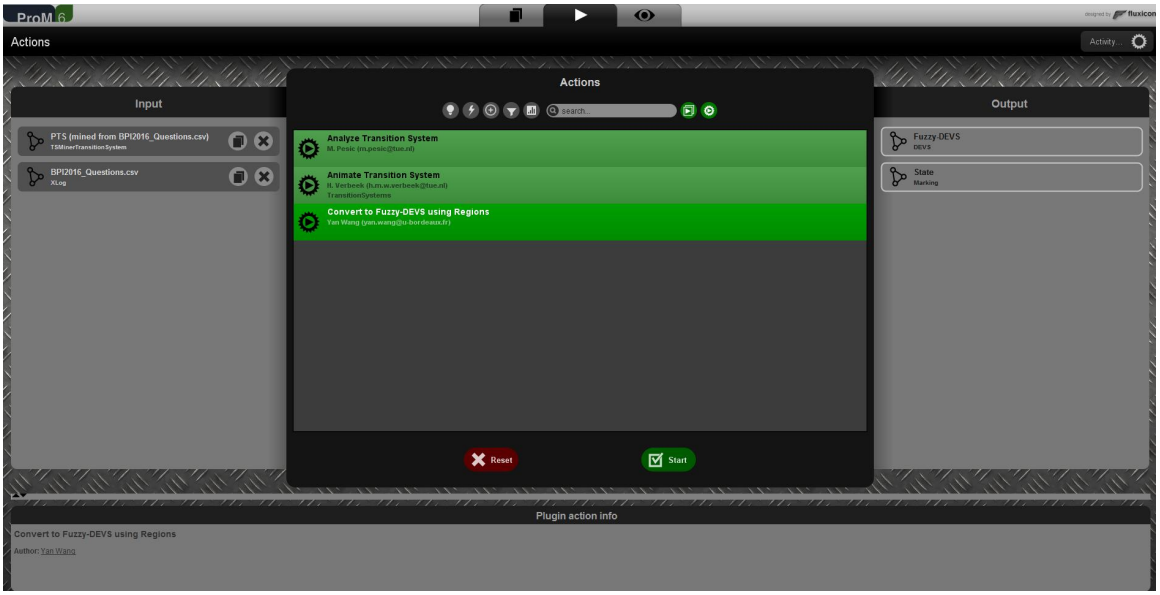


Figure 5.12: The screen of the plugin *Convert to Fuzzy − DEVS using Regions*.

After pressing on the button "start", the Fuzzy-DEVS atomic model is constructed automatically. Figure 5.13 presents the first Fuzzy-DEVS atomic model generated from *Question.csv*. Figure 5.14 shows the corresponding scheme, according to simulation results. As all the activities are the public values in the conceptual structure, all the transitions are converted into the internal transitions defined as the combination of *wm*, ! and the output event, represented as classical arrow. *wm* represents the output port of this atomic model. Behind the graphical notation, the possibility is added. The states are represented by the circle. Every state has a unique identity number. The initial state is converted into the state with label of 1. The output event of the initial state is [ ]. The business process starts from the initial state and continues the transitions automatically until it reaches the desired state.

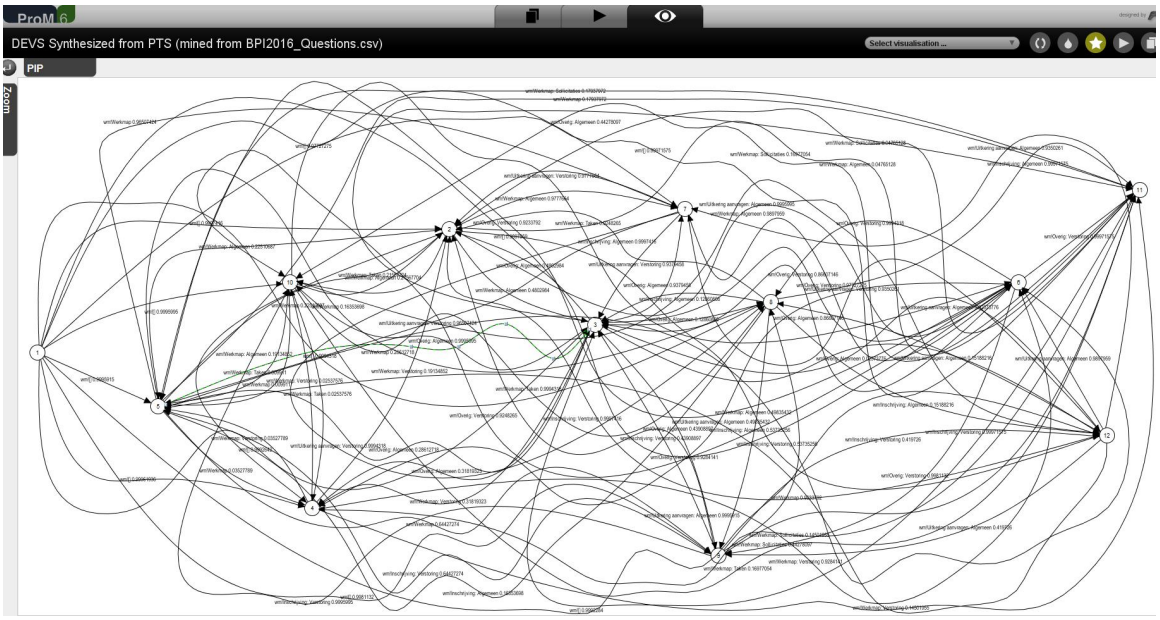


Figure 5.13: Fuzzy-DEVS atomic model generated from *Question.csv*.

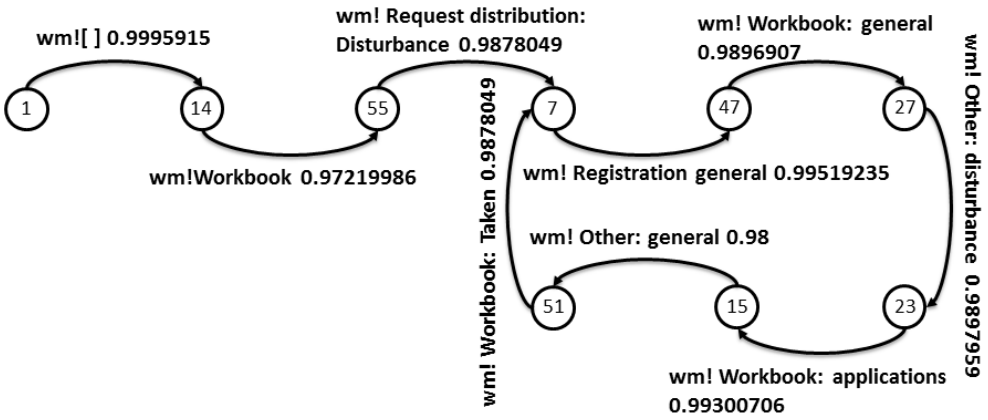


Figure 5.14: Represented scheme from Figure 5.13.

Through the same procedure, we can get the Fuzzy-DEVS atomic model from $Werkmap - message.csv$. The second Fuzzy-DEVS atomic model is generated from $Werkmap - message.csv$ and presented in Figure 5.15. All the transitions are converted into the external transitions defined as the combination of *wm*, ? and input event, represented as diamond arrow. *wm* represents the input port of this atomic model. The initial state is labeled as 1. Every external transition is related to a possibility at the end of the graphical notation. As all the activities are private activities, all the states are set by the infinite time.

The first atomic model is like a generator which is consistently sending outputs. The second atomic model is like a processor which is waiting for events. In order to

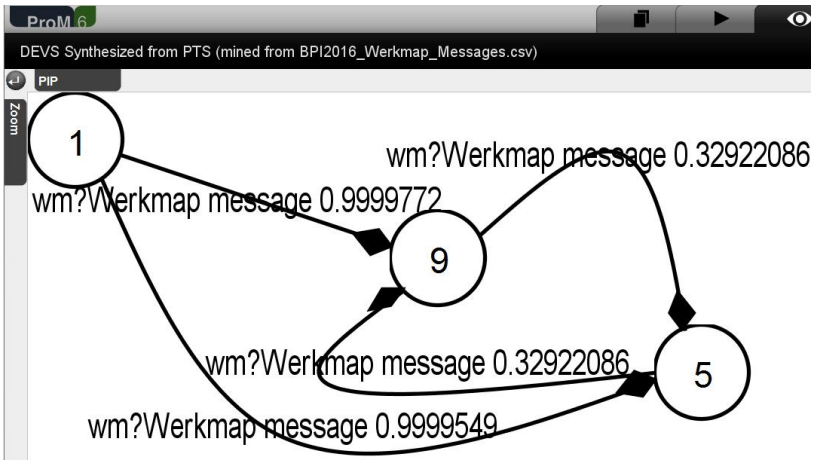


Figure 5.15: Fuzzy-DEVS atomic model generated from $Werkmap - message.csv$.

generate the coupled model, we consider about the port wm for the function $\widetilde{IC}$, to connect two atomic models. Nevertheless, as there is only one function $\widetilde{IC}$ in this case study, $\widetilde{EIC}$, $\widetilde{EOC}$ and $SELECT'$ are not triggered.

Each state of atomic model in Figure 5.13 with a leaving internal transition is given a time life function. This time life function is calculated by AFTC. Figure 5.16 presents one part of the results of the time duration. The initial state is set as the time of 0. This state will immediately go to the next state. In the second state $[Uitkeringaanvragen : Verstoring]$ $(Request\ distribution : disturbance)$, time duration and remaining time are considered as inputs. By applying membership functions, the fuzzy time with five linguistics VS, S, M, B, VB and the fuzzy speed with five linguistics VL, L, A, H, VH are shown. Based on the weighted average method, we get five crisp time and one crisp speed. According to the rule base, the speed is very fast, the final time of $Request\ distribution : disturbance$ is 1122.3 seconds. Through the same calculation of AFTC, we get the final time of $[Werkmap]$ $(Workbook)$ is 2397.69 seconds.

5.3.5 Integrated SimStudio and Its Simulation Results

SimStudio is integrated with the plugin $Convert\ to\ Fuzzy - DEVS\ using\ Regions$. Both of them use JAVA as the programming language. As explained before, SimStudio focuses on either CDEVS or PDEVS. In order to adapt to Fuzzy-DEVS, the possibility and membership coefficient μ is sent to SimStudio. In the atomic model, based on Possibility Measures, SimStudio collects all the internal transitions $\widetilde{\delta}_{int}$ and the external transitions $\widetilde{\delta}_{ext}$ in the same state and selects the one with

```
fuzzy time controller starts
 Time of [[],[]] is 0.0 seconds
fuzzy time controller starts
VS: 0.88 S: 0.0 M: 0.03 B: 0.03 VB: 0.03
VL: 0.82 L: 0.06 A: 0.03 H: 0.03 VH: 0.03
fuzzyspeed: 3234.4312499999996
Selection of speed: very fast
 Time of [[Uitkering aanvragen: Verstoring],[Uitkering aanvragen: Verstoring+]] is 1122.3 seconds
fuzzy time controller starts
VS: 0.99 S: 0.01 M: 0.0 B: 0.0 VB: 0.0
VL: 0.29 L: 0.71 A: 0.0 H: 0.0 VH: 0.0
fuzzyspeed: 3301.161844116149
Selection of speed: fast
 Time of [[Werkmap],[Werkmap+]] is 2397.69 seconds
```

Figure 5.16: Part of fuzzy time results from *Question.csv* by using AFTC.

maximum possibility μ. In the coupled model, based on the function $SELCET'$, SimStudio collects all the external input couplings $\widetilde{EIC}$, the external output couplings $\widetilde{EOC}$ and the internal coupling $\widetilde{IC}$ and selects the one with maximum membership coefficient μ. The crisp final time from AFTC is given to SimStudio for simulation.

```
1 : (the output of Question model) []
Internal transition of Question model: [] & [Inschrijving: Algemeen] ---- 0.9878049
Internal transition of Question model: [] & [Werkmap: Sollicitaties] ---- 0.9897959
Internal transition of Question model: [] & [Uitkering aanvragen: Algemeen] ---- 0.9861111
Internal transition of Question model: [] & [Overig: Algemeen] ---- 0.99300706
Internal transition of Question model: [] & [Overig: Verstoring] ---- 0.9896907
Internal transition of Question model: [] & [Werkmap: Verstoring] ---- 0.9944444
Internal transition of Question model: [] & [Werkmap] ---- 0.9995915
Internal transition of Question model: [] & [Aanmelden en inloggen] ---- 0.975
Internal transition of Question model: [] & [Werkmap: Taken] ---- 0.98
Internal transition of Question model: [] & [Werkmap: Algemeen] ---- 0.99519235
Internal transition of Question model: [] & [Inschrijving: Verstoring] ---- 0.98507464
Internal transition of Question model: [] & [Uitkering aanvragen: Verstoring] ---- 0.9714286
40 : (the output of Question model) [Werkmap]
40 : (the state of Werkmap model) Channel 1
External transition of Werkmap model: [] & Channel 1 ---- 0.9999772
External transition of Werkmap model: [] & Channel 2 ---- 0.9999549
Internal transition of Question model: [Werkmap] & [Werkmap: Taken] ---- 0.9603524
Internal transition of Question model: [Werkmap] & [Uitkering aanvragen: Verstoring] ---- 0.97219986
Internal transition of Question model: [Werkmap] & [Overig: Algemeen] ---- 0.88996136
Internal transition of Question model: [Werkmap] & [Overig: Verstoring] ---- 0.9241352
Internal transition of Question model: [Werkmap] & [Aanmelden en inloggen] ---- 0.9682348
Internal transition of Question model: [Werkmap] & [Werkmap: Algemeen] ---- 0.8436912
Internal transition of Question model: [Werkmap] & [Inschrijving: Verstoring] ---- 0.94709635
Internal transition of Question model: [Werkmap] & [Werkmap: Sollicitaties] ---- 0.9233792
Internal transition of Question model: [Werkmap] & [Inschrijving: Algemeen] ---- 0.9355477
Internal transition of Question model: [Werkmap] & [Werkmap: Verstoring] ---- 0.8633422
58 : (the output of Question model) [Uitkering aanvragen: Verstoring]
Internal transition of Question model: [Uitkering aanvragen: Verstoring] & [Werkmap] ---- 0.97219986
Internal transition of Question model: [Uitkering aanvragen: Verstoring] & [Inschrijving: Algemeen] ---- 0.9878049
Internal transition of Question model: [Uitkering aanvragen: Verstoring] & [Werkmap: Algemeen] ---- 0.71487606
Internal transition of Question model: [Uitkering aanvragen: Verstoring] & [Uitkering aanvragen: Algemeen] ---- 0.3490566
Internal transition of Question model: [Uitkering aanvragen: Verstoring] & [Overig: Algemeen] ---- 0.6101695
```

Figure 5.17: Part of simulation results from *Question.csv* and *Werkmap − message.csv* by SimStudio.

In the case study, part of the simulation results of the coupled model is shown in Figure 5.17. The number before colon represents the time series of minute. The activities or values after colon are output function in the first atomic model and state in the second atomic model. The state shows the number of the channel which is being

used. The time between two time series corresponds to the final crisp time of the state in the first atomic model. When the elapsed time is equal to the time, the internal transition with the maximum possibility executes and sends the output function to the corresponding port wm. The second atomic model receives this output function and execute the external transition with the maximum membership function. Then the state moves to a new state. Based on fuzzy cluster, five values in the first atomic model have membership coefficients with external event $Workbook : message$. In Figure 5.17, the fuzzy time of $Werkmap$ ($Workbook$) is around 39 minutes. After 39 minutes and at the time series of 40, the output function $Workbook$ is sent to the second atomic model to execute the external transition. The membership coefficient $\mu_{\widetilde{IC}}$ between $Workbook$ and $Workbook : message$ is 0.9995915. The state goes to $channel$ 1 with the maximum membership function of external transition 0.9999772 in Figure 5.15. This proves that the activity of $Workbook$ is using channel 1. Table 5.5 summarizes the membership coefficient $\mu_{\widetilde{IC}}$. This $\mu_{\widetilde{IC}}$ changes when the elapsed time e changes. The process of the simulation reveals the critical activities and handles the two goals of this case study.

Table 5.5: The membership coefficient $\mu_{\widetilde{IC}}$ with the elapsed time e

	$Workbook : message$	$Workbook : message$
e	1-40	40-58
$Workbook$	0.9995915	0
$Workbook : general$	0.99519235	0.8436912
$Workbook : application$	0.9897959	0.9233792
$Workbook : taken$	0.98	0.9603524
$Workbook : disturbance$	0.9944444	0.8633422

5.3.6 Case Study of Rabobank Group ICT

In this section, we present the second case study about Rabobank Group ICT. From the description[11], it covers two parts of an IT Service Management (ITSM). These parts are Change Management and Incident Management from the ITIL (Information Technology Infrastructure Library) framework. Rabobank is looking for fact-based insight into sub-questions, concerning the impact of changes in the past, to predict the workload when future changes. One of the goals is to design a predictive model to support Incident Management with less impact of workload at the

[11]https://data.4tu.nl/repository/uuid:c3e5d162-0cfd-4bb0-bd82-af5268819c35

Service Desk and/or IT operations. Thaler et al. (Thaler et al., 2014) take this data and propose an integrated solution to make a detailed analysis of data. The relevant processes (Interaction Management, Incident Management and Change Management) at Rabobank are summarized as follows:

- Interaction Management. In order to manage calls or mails from customers (Rabobank colleagues) at the Service Desk concerning disruptions of ICT services, a Service Desk Agent (SDA) records them in an Interaction and relates them to an Affected Configuration Item (CI). The SDA can either resolve the issue for the customer directly or resolve the service disruption by creating an incident record to assign the issue to an Assignment Group with more technical knowledge. If similar calls/mails are received by the Service Desk, a SDA can decide to relate multiple Interaction records to one Incident record. Then further logging of Activities is done to resolve the service disruption in the Incident record.

- Incident Management. Based on an estimated Impact and Urgency, graded by the SDA, an Incident record is prioritized and limited to resolve the service disruption. A Team Leader within the Assignment Group assigns the records to an Operator. The Operator either resolves the issue for the customer, or reassigns the record to a colleague if some more knowledge are needed. After solving the issue for the customer, the Operator relates the Incident record to the Configuration Item which is caused by the service disruption (Caused By CI). After closing the Incident record, the customer receives an email to be informed that the issue is resolved.

- Change Management. If particular service disruptions happen more often than usual, a problem of investigation is found, conducting an analysis to prevent the happening of the service disruption. The improvement plan leads to a Request for Change (RfC) on the Caused By CI. All CIs are related to a Service Component, Risk Impact Analysis is done by an Implementation Manager assigned to changes which are related to the specific Service Component.

In order to develop corresponding predictive and analysis models, the Rabobank provides four CSV files (*interaction.csv*, *incident.csv*, *incident activity.csv*, *change.csv*) related to these processes. The *incident.csv* and *incident activity.csv* correspond to the goal. Table 5.6 shows the attributes of the two files.

The corresponding descriptions of selected fields are shown as follows:

Table 5.6: The attributes of *incident.csv* and *incident activity.csv*.

Incident	Incident activity
CI Name (aff)	Incident ID
CI Type (aff)	DateStamp
CI Subtype (aff)	IncidentActivityNumber
Service Comp (aff)	IncidentActivity Type
Incident ID	Interaction ID
Status	Assignment Group
	KM number

- CI Name (aff): Configuration Item (CI) where describes an ICT Service. A Service Desk Agent always uses questions in a Knowledge Document (identified by a KM number) to find the correct CI in the Configuration Item Database (CMDB).
- CI Type (aff): Every CI in the CMDB is related to an Entity Type.
- CI Subtype (aff): Every CI in the CMDB is related to a Subtype, which is related to a CI Type.
- Service Comp (aff): Every CI in the CMDB is related to one Service Component, in order to identify the responsible Product Manager. A Service Component is equal to a product in the Bill of Material and is part of Services.
- KM number: A Knowledge Document contains default attribute values for the Interaction record. There are also a set of questions for a Service Desk Agent to derive Configuration Item. This document can determine Impact and Urgency for the customer.

The relationship of these two files is that *incident activity.csv* is the aspect of *incident.csv* so we focus on the previous one. $IncidentID$ is identified as the reference of the event data. We also create a new attribute named *DataStampStart* which is taken from the $OpenTime$ as the first time of each incident and we identity *DateStamp* as the next time of each incident. The conceptual structure of *incident activity.csv* is shown in Figure 5.18. The incident has several attributes but we focus on the property of $IncidentActivity\ Type$. Moreover, the values *Assignment*, *Communication with customer*, *Communication with vendor*, *External vendor assignment*, *External vendor reassignment* and *Resolved* are selected as the interesting public values. The attributes of *DataStampStart* and *DataStamp* are selected

as start timestamp and end timestamp. *IncidentID* is selected as trace in the event logs.

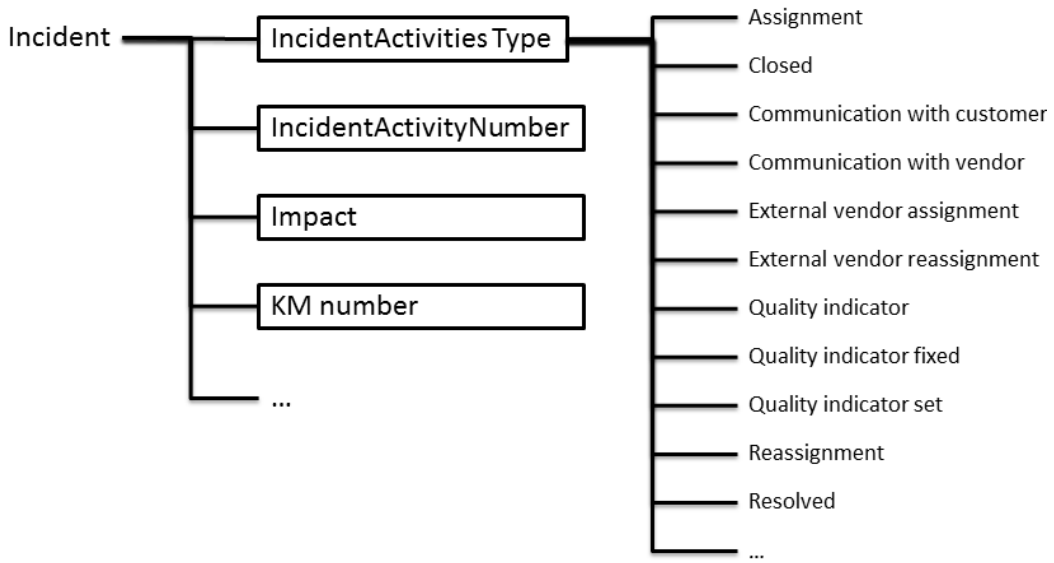


Figure 5.18: The conceptual structure of *incident activity.csv*.

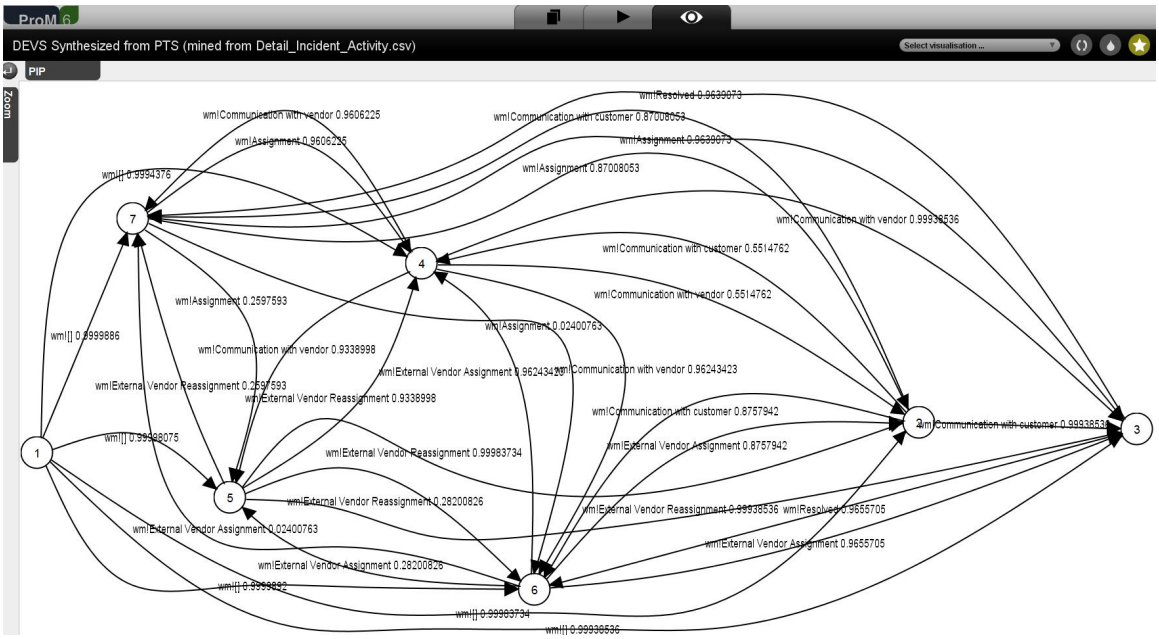


Figure 5.19: Fuzzy-DEVS atomic model generated from *incident activity.csv*.

By following the application of D2FD method and executing three plugins, a Fuzzy-DEVS model is mined as shown in Figure 5.19. The state starts from the initial state 1 until it reaches the end state 3. All the transitions are internal transitions represented as classical arrow. The graphical notation of the internal transition is

```
1 : []
Internal transition: [] & [Assignment] ---- 0.9999886
Internal transition: [] & [External Vendor Reassignment] ---- 0.97727275
Internal transition: [] & [Communication with vendor] ---- 0.9994376
Internal transition: [] & [External Vendor Assignment] ---- 0.9997704
Internal transition: [] & [Resolved] ---- 0.99938536
Internal transition: [] & [Communication with customer] ---- 0.99983734
785 : [Assignment]
Internal transition: [Assignment] & [Communication with customer] ---- 0.87008053
Internal transition: [Assignment] & [External Vendor Reassignment] ---- 0.9990174
Internal transition: [Assignment] & [External Vendor Assignment] ---- 0.90621066
Internal transition: [Assignment] & [Resolved] ---- 0.9639073
Internal transition: [Assignment] & [Communication with vendor] ---- 0.9606225
1439 : [External Vendor Reassignment]
Internal transition: [External Vendor Reassignment] & [Resolved] ---- 0.99938536
Internal transition: [External Vendor Reassignment] & [External Vendor Assignment] ---- 0.9802183
Internal transition: [External Vendor Reassignment] & [Communication with customer] ---- 0.99983734
Internal transition: [External Vendor Reassignment] & [Assignment] ---- 0.9990174
Internal transition: [External Vendor Reassignment] & [Communication with vendor] ---- 0.9522241
1597 : [Communication with customer]
Internal transition: [Communication with customer] & [External Vendor Assignment] ---- 0.17080835
Internal transition: [Communication with customer] & [Communication with vendor] ---- 0.5514762
Internal transition: [Communication with customer] & [Resolved] ---- 0.99938536
Internal transition: [Communication with customer] & [Assignment] ---- 0.87008053
1772 : [Resolved]
Internal transition: [Resolved] & [Assignment] ---- 0.9639073
Internal transition: [Resolved] & [External Vendor Assignment] ---- 0.456111
```

Figure 5.20: Part of simulation results from *incident activity.csv* by SimStudio.

combined with output port *wm*, !, output function and membership function. Every state has a final crisp time which is calculated from AFTC. Figure 5.20 presents the simulation results of this model. The number before colon represents the time series of hours. The activities after colon are output function. The time between two time series corresponds to the final crisp time of the state. When the elapsed time is equal to this time, the internal transition with the maximum membership function is triggered and sends the output function to the output port *wm*. For example, the first output function is [ ] which has a time of 0. The internal transition from [ ] to *Assignment* has the maximum membership function. After the time of *Assignment* 784 hours, the output function *Assignment* is sent to the port *wm* at the time series of 785. Again after the time 654 hours, the output function *External vendor reassignment* is sent to the port *wm* at the time series of 1439. The simulation results illustrate the critical workload based on goals.

5.4 Conclusion

The D2FD method is implemented by the plugin on ProM. ProM is extremely strong which is composed by hundreds of plugins. It is easy for process mining users to implement their own plugin with algorithm and techniques on the platform of

ProM. Compared with other process mining tools, ProM provides more functionality than some of the less mature tools. The simulation of Fuzzy-DEVS model in the D2FD method is implemented by SimStudio. SimStudio is easy to be integrated in the plugin on ProM. The simulation results can be visualized by the support of ProM. Two case studies illustrate the feasibility of this tool. Although the data of two case studies are quite big and are not completely analyzed, the interesting simulation results by using the proposed plugin are able to solve business problem and reveal optimal business processes. This plugin is able to identify the underlying relationships and make model visual. The identified relationships and the fuzzy cluster can make the complex and separated data connected each other.

From the results of two case studies, we can get only one result with list of events from the simulation of each models. Some more information, for example membership function, are still displayed on the model but cannot be observed by simulation. In the following chapter, a replicative method and a predictive method are proposed for the validation of Fuzzy-DEVS model in the D2FD method. These method can provide more simulation results from the same models.

Chapter 6

Validation of D2FD Method

6.1 Introduction

Model verification is often defined as ensuring that the computer program of the computerized model and its implementation are correct. Model validation is usually defined as substantiation that a computerized model within its domain of applicability possesses a satisfactory range of accuracy consistent with the intended application of the model (Schlesinger, 1979). Model validation is important as the last step of D2FD method. However, the discovered Fuzzy-DEVS model in the D2FD method can get only one simulation result, and this result is not always validated based on different enterprise requirements. In this chapter, we propose a morphism-based model approximation method and a predictive method using Granger Causality for the validation of D2FD method. This chapter is organized as follows. Section 6.2 gives the background of Model Morphism (MoMo), verification and validation methods in the modeling and simulation and the algorithm of Granger Causality. Section 6.3 explains the two proposed methods. Later we continue to use the real case study of dutch employee insurance agency to support these two methods. The results of this case study is validated by comparing to other models. The conclusion is given in section 6.5.

6.2 Background

Morphism-based model approximation and the predictive method are based on the use of MoMo and the algorithm of Granger Causality. In this section, we give the background of these two parts. In addition, the verification and validation methods in the theory of modeling and simulation are reviewed. These methods can be

applied after generating a new model from the two proposed methods. The two proposed methods are also related to replicative and predictive validation methods. A new paradigm of inferring models and simulation between the real world and the simulation world is proposed.

6.2.1 Model Morphism(MoMo)

In the mathematical field, morphism is defined as a structure-preserving map between two mathematical structures. When applying this term in the modeling and simulation, MoMo proposes a terminology and adapts it into the structure of models. According to the hierarchy of system specification in Table 2.2, a morphism is a relation which connects the pair of systems at each level of the hierarchy. It specifies the relationships between two or more models described in the same or different languages. The term morphism is only used in the information and communication technologies. Agostinho et al. (Agostinho et al., 2007) propose a model driven architecture and MoMo method to solve the enterprise interoperability problem. The morphism is used to transform the model from one language to another. Agostinho et al. (Agostinho et al., 2011) also propose the integration of trace ability functionalities to solve the problem of sustainability in enterprise interoperability.

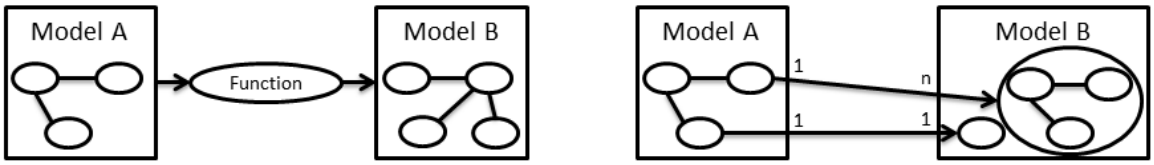


Figure 6.1: The classification of MoMo.

The morphism can be seen as the operator and the model as the operand. According to the operation, MoMo can be distinguished between model-altering and non-altering (InterOP, 2005). Model-altering modifies the operand. Conversely, non-altering does not modify the operand. As depicted in Figure 6.1 (a), the source model A identified as the operand is transformed by some functions into the target model B. These functions identified as operators apply several rules for the transformation. The MoMo group of the InterOp project[1] makes a formal definition of model-altering:

Let MOD be the set of all multi-graphs that are the representation of some models in some language. If there is a model $A \in MOD$ and a function $\tau : MOD \rightarrow MOD$, a model-altering morphism is τ, having $\tau(A) = B$, and $B \in MOD$.

[1] www.interop-noe.org

As depicted in Figure 6.1 (b), non-altering morphism is close to the tradition concept of model mappings. Compared with model-altering, there is no changes between the source model A and the target model B. The relationship for example similarity (1 to 1) and composition (1 to n) is identified between these two models. The MoMo group of the InterOp project also makes a formal definition of non-altering:

Let MOD be the set of all multi-graphs that are the representation of some models in some language. If there are at least two models $A, B \in MOD$, a non-altering morphism is a relation τ', having $\tau' \subseteq Sub(A) \times Sub(B)$, where $Sub(X)$ is a sub-graph of X.

In addition, MoMo can be extended by the ontology to describe the manipulation of models. As the ontology provides a valuable knowledge-based techniques about methods, decisions and suggestions, MoMo ontology is able to provide more solutions for enterprise interoperability problems.

6.2.2 Verification and Validation of Modeling and Simulation

In general, the process of developing models and simulations are based on a set of fundamental assumptions (Pace, 2000). Conversely, D2FD method proposes a process of inferring models and simulations. Compared with the paradigm of Sargent (Sargent, 2009), a corresponding paradigm is shown in Figure 6.2. There are a Real World and a Simulation World. In the Real World, there exist some systems or problems. System problem refers to the data level of system knowledge. System data is obtained by conducting experiments on the system. In the simulation world, the inference model is the mathematical discipline to represent the system. It is mined from system in the real world. The simulation model is the inference model running on the computer system like ProM and applying simulation model specification. The simulation model data and results are the data and results from experiments conducted on the simulation model. Specification verification is used to check whether the software design is adapted on the specified computer system. Implementation verification is defined as assuring that the simulation model conforms to the specification. Operational validation is defined as determining that the output behavior of the model has sufficient accuracy for the intended purpose of the model. In this thesis, assuming that the verification of specification and implementation are completed, we focus on the validation between the simulation model results and real world.

The operational validation is important for D2FD method. The design of a model that appeared complete and robust can become incoherent, incomplete and potentially invalid during simulation implementation. Sargent (Sargent, 2009) not

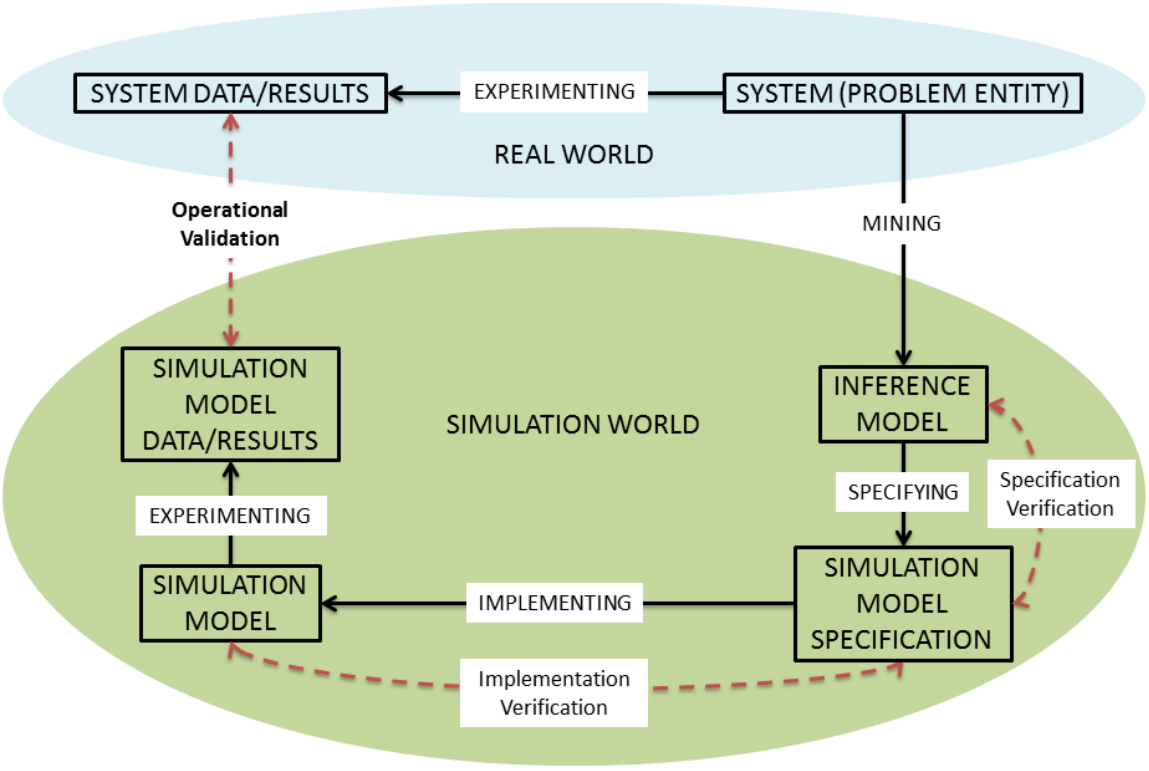


Figure 6.2: The structure of the inferring process with verification and validation.

only proposes a paradigm that relates verification and validation to the model development process but also presents various techniques for building valid and credible simulation models. A combination of these techniques is generally used. These techniques are shown as follows:

- Animation: As the time elapse, the behavior of the model is displayed graphically.
- Comparison to Other Models: Make a comparison between the models which have already been validated and the models which are anticipated to be validated.
- Degenerate Tests: The degeneracy of the behavior of the model is tested by selecting appropriate of the input and internal parameters.
- Event Validity: Make a comparison of the occurrences of events between the simulation model and the real system.
- Extreme Condition Test: For any extreme and unlikely combination of levels of factors in the system, the model structure and outputs should be plausible.

- Face Validity: Make validation of the model and its behavior directly from individuals who know about the system.
- Historical Data Validity: If the part of the historical data is used to build the model, the other part of historical data can be used to check whether the behaviors of model are validated.
- Historical Methods: There are three historical methods of validation which are rationalism, empiricism, and positive economics. Rationalism requires the model to be validated by using logic deduction from the assumptions which everyone knows. Empiricism requires every assumption and outcome of the model to be empirically validated. Positive economics requires the model which can only predict the future.
- Multistage Validity: The three historical methods of rationalism, empiricism, and positive economic are combined together for validation.
- Internal Validity: Several replications (runs) of a stochastic model are validated by the amount of (internal) stochastic variability in the model.
- Operational Graphics: Values of various performance measures are shown graphically as the time elapse.
- Parameter Variability - Sensitivity Analysis: The behavior and output of the model are validated by changing the values of the input and internal parameters.
- Predictive Validation: Make a comparison between the system model and the predicted model.
- Traces: The logic and accuracy of the model are validated by tracing the behavior of different types of specific entities.
- Turing Tests: Individuals who know about the operations of the system models make tests of discriminating between system and model outputs.

Some more validation methods in the theory of modeling and simulation were studied. Elzas (Elzas, 1984) starts to talk about two kind of modeling ways top-down and bottom-up. In addition, a system analysis validation framework and a system design validation framework are shown. Law et al. (Law, 2008) propose the whole steps for building valid and credible simulation models. They also discuss the difficulties in using these techniques for validation. Gore et al. (Gore and Diallo,

2013) provide the approach used in practice with a formal specification languages. They force attention to mathematical details. There are also some other verification techniques in modeling and simulation areas (Gajski et al., 2009; Harbola et al., 2012).

Validity corresponds to the replicative, predictive and structural validity (Zeigler et al., 2000). The experimental frame is critical for validation because it provides the conditions to make experiments with both the model and the system. Replicative refers to quantitative comparison and qualitative comparison. This comparison is using trajectories under this experimental frame. Quantitative comparison requires a metric and a tolerance. The metric provides a numerical basis which calculates the goodness of fit. The tolerance makes a examination whether the fit is good enough. If the fit is over the tolerance, the model is not enough to be qualified as valid. The qualitative comparison contains two methods which are visualization and animation. The visualization tries to translate complex data into graphical structure that human can understand. Animation simulates the behavior of the model as the model moves through time. In the predictive validation, the initial state of the model can be inferred from the past system observations. Structural validity refers to cross-model validation for example between models at different levels of resolution.

6.2.3 Granger Causality

The general philosophical definition of Causality(Bunge, 2017) is "the natural or worldly agency or efficacy that connects one process (the cause) with another process or state (the effect), where the first is partly responsible for the second, and the second is partly dependent on the first." Granger Causality (G-causality) as one of the causality measures is proposed by Granger in 1969 (Granger, 1969). In terms of linear regression modeling, Granger Causality as a statistical formalization is defined that a time series X Granger-causes Y if the inclusion of past observations of X helps reduce the prediction error of Y. The information of X is better than the information already in the past of Y as well as in the past of other variables Z. G-causality is mostly implemented via linear vector autoregressive modelling of timeseries data (Geweke, 1984; Seth, 2010a). Some other G-causality can use nonlinear, time-varying, and non-parametric models (Roebroeck et al., 2011; Dhamala et al., 2008). To illustrate G-causality, we use two time series $X_1(t)$ and $X_2(t)$ (both of length T) which can be described by a bivariate autoregressive model (Seth, 2010b):

$$X_1(t) = \sum_{j=1}^{p} A_{11,j} X_1(t-j) + \sum_{j=1}^{p} A_{12,j} X_2(t-j) + \xi_1(t) \tag{6.1}$$

$$X_2(t) = \sum_{j=1}^{p} A_{21,j} X_1(t-j) + \sum_{j=1}^{p} A_{22,j} X_2(t-j) + \xi_2(t) \tag{6.2}$$

Where p is the maximum number of lagged observations in the model and $p < T$, A represents the coefficients of the model, ξ_1, ξ_2 are the prediction errors for each time series. If the variance of ξ_1 (or ξ_2) is reduced by the inclusion of the X_2 (or X_1) terms in the equation 6.1 (or 6.2), then it is said that X_2 (or X_1) G-causes X_1 (or X_2). Assuming that X_1 and X_2 are covariance stationarity for example mean and variance are fixed, the magnitude of the interaction between them can be measured by the log ratio of the prediction error variances based on the restricted R and unrestricted U models as well as G-causality (Seth, 2007):

$$gc_{2\to1} = log\frac{var(\xi_{1R(12)})}{var(\xi_{1U})} \tag{6.3}$$

Where $\xi_{1R(12)}$ id derived from the model omitting all the $A_{12,j}$ coefficients in the equation 6.1, and ξ_{1U} is derived from the full model. In addition, G-causality can be extended to the multivariate case by a Taylor expansion in which the G-causality of X_1 is tested in the context of multiple variables X_1, ..., X_N(Seth, 2007).

In addition, G-causality can extend one of the variables based on the quantification of the statistical autonomy. In this case, a variable X_1 is G-autonomous in which the prediction error of X_1 is reduced by the inclusion of its own past. According to equation 6.1 and 6.2, the G-autonomous of X_1 is given by:

$$ga_{X_1|X_2} = log\frac{var(\xi_{1R(11)})}{var(\xi_{1U})} \tag{6.4}$$

Where $\xi_{1R(11)}$ id derived from the model omitting all the $A_{11,j}$ coefficients in the equation 6.1, and ξ_{1U} is derived from the full model.

By combining G-causality and G-autonomy, G-emergence is proposed in terms of weak emergence (Seth, 2010b). An emergent property is somehow "more than the sum" of its component parts. Weak emergence defines that a macro-level property is weakly derived from the interaction of micro-level components. A macro-variable M is G-emergent from a set of micro-variables m so G-emergence can be defined as:

$$ge_{M|m} = ga_{M|m}(\frac{1}{N}\sum_{i=1}^{N} gc_{m_i\to M}) \tag{6.5}$$

Where M is both G-autonomous with respect to m and G-caused by m. Especially, $ge_{M|m}$ will be zero either if M is independent of m or if M is fully predicted by m.

6.3 Two Proposed Methods for Model Validation

D2FD method provides a system inference method which mines always the same Fuzzy-DEVS model from one event logs. For this reason, we propose two methods to improve the interoperablity of the D2FD method. Morphism-based model approximation provides a model-altering morphism with the modification of functions. The predictive method is integrated with G-causality. These two methods are evaluated by the case study of dutch employee insurance agency and validated by comparing to other models.

6.3.1 Morphism-Based Model Approximation Method

In first stage of the D2FD method, the main idea of identifying process instance is to select one final property among several ones then the values corresponding to this property can be transformed in the event logs as explained in section 3.3.4. However, sometimes the selected process instances cannot meet the requirements and the discovered model is too complicated to validate. In the AFTC as explained in section 4.2.3, the form of membership functions and rule base is designed based on assumptions or hypothesis which are disconnected to real system. The discovered Fuzzy-DEVS model, in which each transition and coupling has a corresponding possibility, provides more information than the classical DEVS.

The enterprise can have different requirements. On the one hand, the model needs to be simple and visible so it is easy to be analyzed and validated. On the another hand, single model cannot satisfy different enterprise requirements. Hence, morphism-based model approximation is proposed to reconstruct a new model closer to the enterprise requirements. According to the problem in section 1.2.3, there is a technological barrier caused by the use of different methods and techniques to represent information. Morphism-based model approximation is able to provide an integrated approach for validation. In the D2FD method, this integrated method can be applied on reducing the scope of the functions. Based on requirements, an appropriate percentage is given to the reduction rate. This method provides a model-altering way which modifies three functions:

- The amount of references related to events in event data.
- All the possibilities μ in Fuzzy-DEVS atomic model $\widetilde{D}$.
- The membership functions and rule base in AFTC.

The events are described by references. If the amount of references is reduced by the reduction rate, the quantity of events decreases. The possibility μ consists of fuzzy internal transition μ_{int}, fuzzy external transition μ_{ext}, fuzzy external input coupling μ_{EIC}, fuzzy external output coupling μ_{EOC}, fuzzy internal coupling μ_{IC}. μ is always between 0 and 1. A threshold from the reduction rate can be set on this possibility. The percentage in the membership functions and rule base can also be changed from reduction rate.

6.3.2 Predictive Method Using Granger Causality

Recall the G-causality, a time series X Granger-causes Y if the inclusion of past observations of X helps reduce the prediction error of Y. Moreover, Y can be G-autonomous by its own variable. Both G-causality and G-autonomy compose G-emergence. If we apply this in the process model, the trigger of each transition can be G-caused by the past transitions and states and G-autonomous by itself. In the D2FD method, the trigger of fuzzy internal transition and fuzzy external transition depends on the possibility $\mu_i nt$ and $\mu_e xt$. These possibilities are calculated by the frequency F in the equation 4.8. On the one hand, Dependency Method illustrates that the possibility of the transition can be G-autonomous by its own frequency. On the another hand, as the model simulates through the time, the possibility of each transition can be predicted by the past frequencies. This predictive method is shown by the following equation:

$$\mu_p(F) = a_1 \frac{1}{N} \sum_{i=1}^{N} \mu(F_i) + a_2 \mu(F) \tag{6.6}$$

Where N is the memory depth and a_1, a_2 are the memory effects. F measures the frequency of the transition from event logs. F_i measures the frequency of the existing simulated transitions in the depth of i. The memory depth defines the number of simulated states from the current to the past. It is important to set a smaller number for the memory depth when there is a big amount of the states in the model. The memory effects can be tested in three conditions. The condition M defines the predicted possibility $\mu_p(F)$ is both based on G-causality and G-autonomous ($a_M = [0.5, 0.5]$). The condition H defines $\mu_p(F)$ focuses more on the past observations of frequencies ($a_H = [0.8, 0.2]$). The condition L defines $\mu_p(F)$ focuses more on the current frequency ($a_L = [0.2, 0.8]$).

In order to better explain F_i in equation 6.6, we use a small example here. Imagine that we have a set of states $s_0, s_1, ..., s_n$, the simulation process is $s_0 \rightarrow s_1 \rightarrow$

$s_2 \rightarrow s_3 \rightarrow s_4 \rightarrow s_5$. If the current state is s_4 and we want to calculate the possibility of $\mu_p(s_4 \rightarrow s_5)$, F_1 is the frequency which contains only $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3$, F_2 is the frequency which contains only $s_0 \rightarrow s_1 \rightarrow s_2$, F_3 is the frequency which contains only $s_0 \rightarrow s_1$, F_4 is the same as F so the maximum memory depth N is 3.

6.3.3 Case Study Relevant to Two Methods

In this section, we continue to use the case study in section 5.3.1. As explained in the Figure 5.14, SimStudio can figure out a sequence of simulation results. These simulation results are used to solve the two main goals and they can be represented as a statistical graph as shown in Figure 6.3. This sequence is shown as follows: *Workbook* takes 40 minutes, *Request distribution* : *disturbance* takes 18 minutes, *Workbook* : *general* takes 44 minutes, *Other disturbance* takes 10 minutes, *Workbook* : *application* takes 40 minutes, *Other general* takes 44 minutes, *Workbook* : *taken* takes 40 minutes.

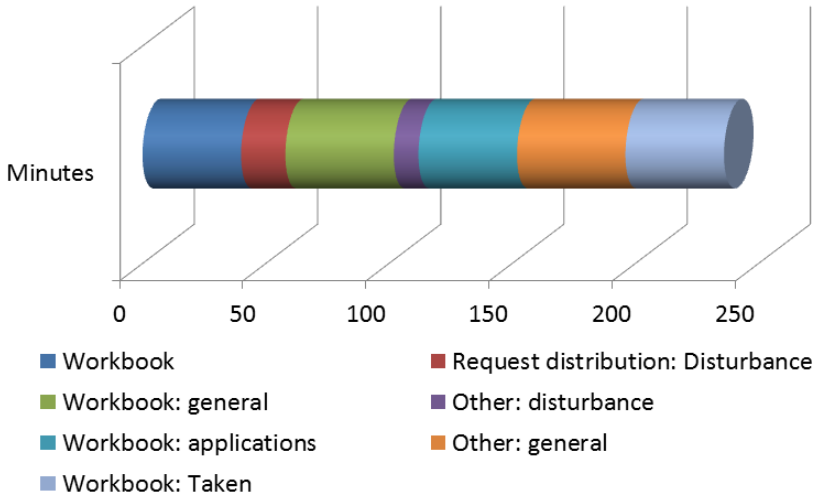


Figure 6.3: Represented statistical graph from the simulation results in Figure 5.14.

Since we can get only one simulation result from the case study, we use morphism-based model approximation to obtain a new model and capture some other simulation results. As an example to apply morphism-based model approximation, we change the threshold of possibility from 1 to 0.9995. Therefore, there are less transitions than the transitions in the model as shown in Figure 5.13. Figure 6.4 shows the corresponding simulation results with the new threshold. *Other general* takes 44 minutes, *Workbook* takes 40 minutes, *Request distribution* : *disturbance* takes 18 minutes, *Registration general* takes 43 minutes.

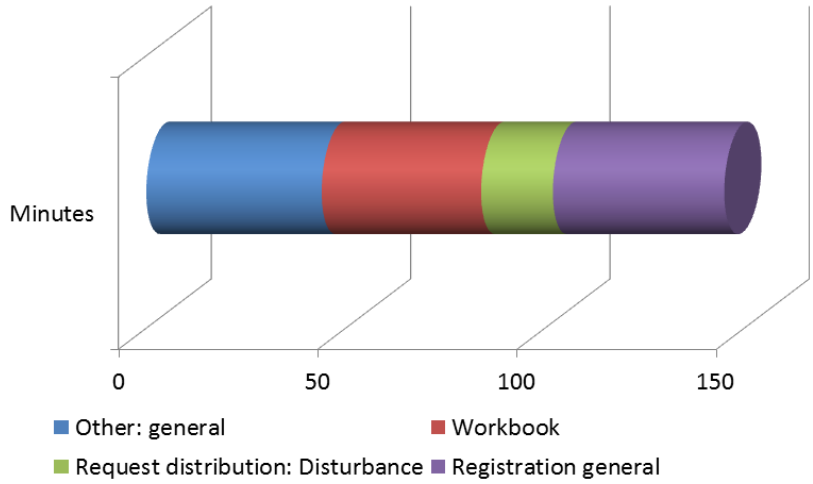


Figure 6.4: Represented scheme of simulation results by reducing the possibility from 1 to 0.9995.

Moreover, we can use G-causality method to get some predictive simulation results. According to equation 6.6, we set the memory depth N as 2 and the memory effects $a1$ as 1 and $a2$ as 2. The condition is H. Figure 6.5 presents the new simulation results based on memory depth of 2. *Request distribution* : *disturbance* takes 18 minutes, *Request distribution* : *general* takes 43 minutes. If we change the memory depth N to 4, we are able to get another simulation results as shown in Figure 6.6. *Other general* takes 44 minutes, $Workbook$: *general* takes 44 minutes, $Workbook$: *disturbance* takes 40 minutes. As a result, we can find out that both memory depth and memory effects influence the final simulation results.

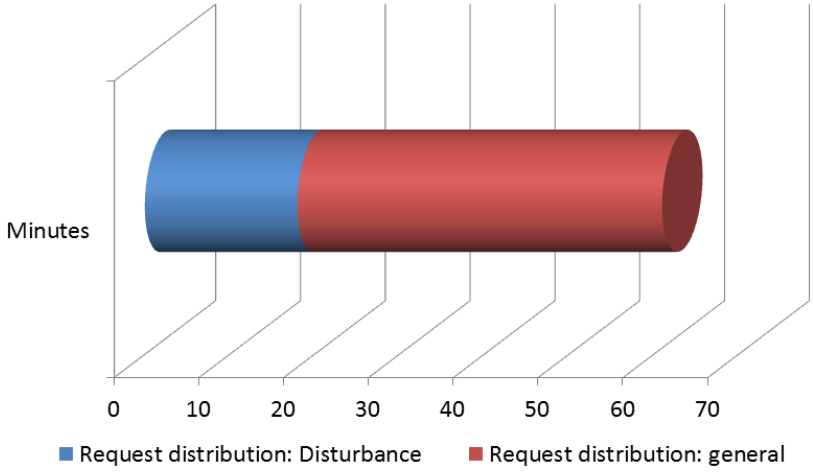


Figure 6.5: Represented scheme of simulation results by setting memory depth as 2 and setting the condition H.

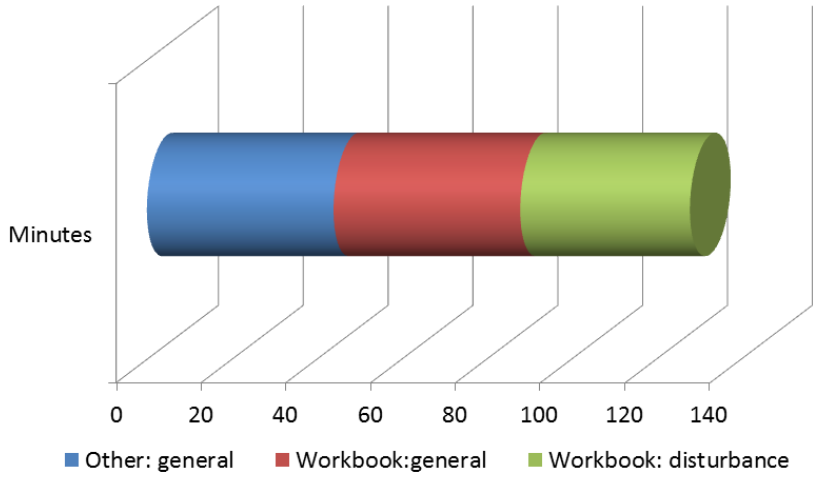


Figure 6.6: Represented scheme of simulation results by setting memory depth as 4 and setting the condition H.

6.3.4 Validation of Case Study of Dutch Employee Insurance Agency

According to section 6.2.2, it is difficult to make face validation. So we choose to compare with other models for the validation of the case study of Dutch employee insurance agency. The compared models come from the study of Jalali (Jalali, 2016). Figure 6.7 shows the compared model generated from *Question.csv*. This model is constructed by using another process mining tool Disco (Van der Aalst, 2011). As it can be seen from this process, there are three pages including the home page which customer visits more than others. Then customers visit both *mijn_werkmap* and *taken* pages represent *workbook* and *taken* pages. The two pages have bigger frequency of 5,887 and 13,696 than others. Compared between the model in Figure 5.14 and in Figure 6.7, both models have the mainstream process from $Workbook$ to $Workbook : taken$. Moreover, Fuzzy-DEVS model gives more simulation results between $Workbook$ and $Workbook : taken$.

Another comparison happens based on $Werkmap - message.csv$. Figure 6.8 shows the compared model by Disco. The channel has no hierarchy, so the members are appeared in the mined model, i.e. channel 1 and channel 2. The Complain Type hierarchy is explored at its highest level of granularity. In this process, channel 1 and 2 are used to issue a complain. The channel 1 is used more than the other one. There are also some cases illustrating that a user switched between these channels. When comparing with the Fuzzy-DEVS model in Figure 5.15, these two models show almost the same process between the channels. However, from the simulation results

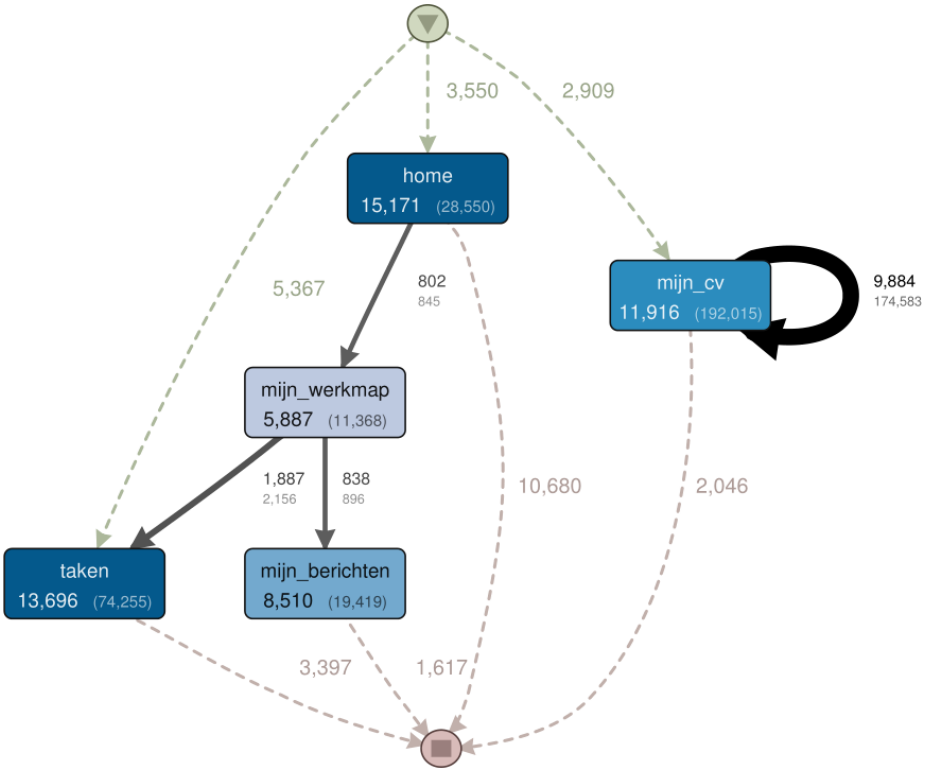


Figure 6.7: Compared model from *Question.csv*.

of Fuzzy-DEVS model, we are able to get the connection between activity and channel. For example, the activity of *Workbook* is using channel 1 in Figure 5.17.

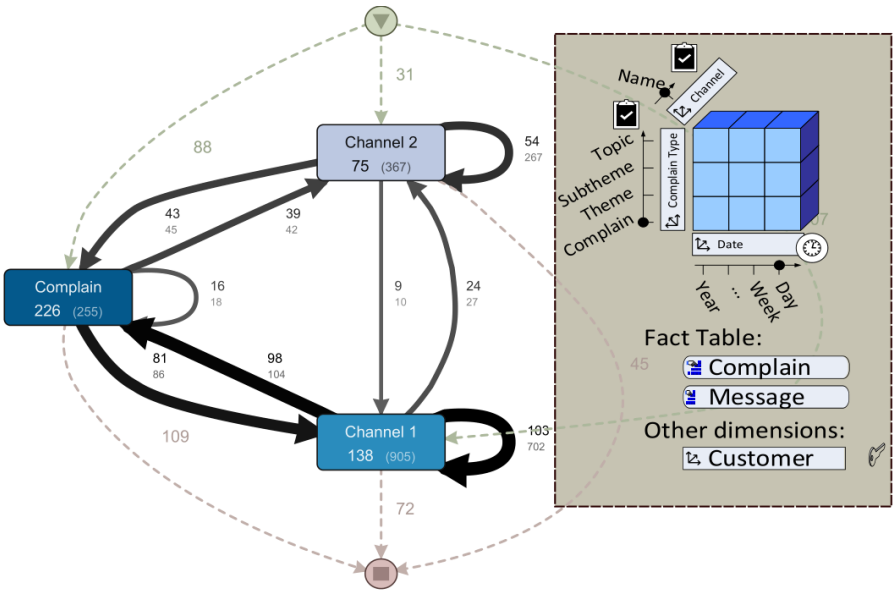


Figure 6.8: Compared model from *Werkmap – message.csv*.

6.4 Conclusion

Validation is very important for discrete event simulation models. The aim is to prove the coherence of the model, to ensure the correct use of the modeling methods and account for the description of the requirements that prevailed in the existence of models. Morphism-based model approximation proposes a model-altering method. Three functions in the D2FD methods provides flexible solutions to handle the problems based on enterprise requirements. Model Morphism is able to specify the relationships between two or more model described in the same or different languages. Predictive method proposes to integrate G-causality with the possibility of fuzzy internal transition and fuzzy external transition. The original Dependency Method is extended with G-causality to be G-emergency. A new nonlinear predictive model is generated by applying this G-emergency method. The proposed integrated methods applies the concept of MoMo and the algorithm of Granger Causality to modify the functions in D2FD method. By applying these two methods, the case study is able to get some new and interesting model and simulation results. Compared with the other model in the previous studies, the model from D2FD method is validated. Later in the conclusion, the limitations of D2FD method are discussed, and the perspectives are proposed based on these limitations.

General Conclusion

Conclusion We propose a D2FD method, as presented in Figure 1.2, an integrated approach to the discovery of discrete event simulation model from real data. In particular, we mine a Fuzzy-DEVS model from event data. This method provides a relatively complete solution for system inference. The process discovery techniques are extended in process mining. Thanks to fuzzy logic, D2FD method is able to represent imprecision from event data and distinguish the discovered model what is important and what is not. Compared with the state of the art, D2FD method has better performance as follows:

- Five-step method is proposed to extract event logs from event data;
- Frequency is taken into account by using dependency method;
- AFTC is proposed to handle time aspects;
- Fuzzy cluster is applied to discover coupled model, to handle modularity.

The five-step method deals with the challenge of collecting data in process mining. The conceptual structure inherited from SES is used to build a modular and hierarchical abstraction from the collected data. The underlying relationships can be observed from this structure. These relationships help to focus on the interesting part of event data and discover the connections between documents. The general mapping connects event data to event logs. D2FD method proposes an improved region-based approach which is more appropriate to discover the timing function of Fuzzy-DEVS model. The underlying relationships help to discover internal transition and external transition of Fuzzy-DEVS model. The discovered Fuzzy-DEVS can be the extension of process model in process mining, which provides modularity. And D2FD method takes frequency and time into consideration. The Dependency Method is used to extract the frequency of events from event logs, and the AFTC is used to extract the timing aspects. These methods expand rough approximations into fuzzy environment and solve issue of imprecision. Thanks to fuzzy cluster, Fuzzy-DEVS atomic models

are coupled together to represent hierarchical structure of complex system. In order to validate the final mined model, two approaches are proposed:

- Morphism-based model approximation is proposed to handle replicative validity;
- Granger causality is applied to handle predictive validity.

Morphism-based model approximation can rebuild a new model and makes the simulation results closer to accuracy. In the simulation process, the execution of transitions not only can be predicted by themselves using dependency method, but also can be predicted by their previous transitions using G-Causality. Two proposed methods provide integrated approaches concerning processes to overcome the technological barriers. The new mined model is able to obtain different simulations results associated with enterprise requirements.

The D2FD method is implemented as plugin on ProM, which is practical, visible, automatic and available. The resulting model is simulated by using SimStudio. There are two case studies discussed in this thesis, which illustrates the feasibility of the plugin of D2FD method. In particular, the results of the first case study from dutch employee insurance agency not only evaluates two validation approaches but also can be validated by comparing to other studies. Although the data of two case studies are quite big and are not completely analyzed, the interesting simulation results by using the proposed plugin are able to solve business problem and reveal optimal business processes. This plugin of D2FD method is able to identify the underlying variables relationships and make model visual. The identified relationships and the fuzzy cluster can make the complex and separated data connected each other.

Limitations D2FD method provides a system inference method using process mining techniques. This method is able to discover a Fuzzy-DEVS model from the event data. In addition, two methods are proposed for the validation of D2FD method in chapter 6. However, there are still some limitations and shortcomings in this method. In general, there are four main limitations in the D2FD method as follows:

- Threshold of Event Data: The definition of event data is explained in section 3.2 with the 16 guidelines. They make a general standard for the event data to be selected. However, according to section 1.2.1, only part of problem has been solved. Until now, how to get event data coming from ERP, internet of events and so on is still a big challenge in the process mining.

- Over-fitting: In the second phase of region-based approach, we construct regions from TS while we are discovering concurrency. TS and Fuzzy-DEVS are equivalent from a behavioral point of view. So this may lead to over-fitting and Fuzzy-DEVS is not generalizing. An over-fitting model is defined that the next trace which we will observe will actually not fit into the model. The model has a poor predictive power.

- Last activity: The execution of last activity (value) is not taken into account in the D2FD method. In the toy case of Figure 4.6, in terms of $L_{end} = [(S, E)^5, (E, S)^8]$, S and E as public values in the state $S9$ cannot be triggered. The state of $S9$ is infinite so there is no output function sent to other models. When the events in event logs are messy and complex, the last state can have internal transitions. The problem happens when there is no internal transitions and the value inside is public value.

- Ontological Alignment: As explained in section 4.2.4, the conditions to form clusters depend on two variables which have the same name or one belongs to one part of another. Ontological alignment which provides some methods in the level of knowledge and semantic can make a better choice for the clusters.

Besides, there are some other limitations when applying D2FD method in the case study. The case studies in section 5.3.1 and 5.3.6 do not consider about the whole problems from enterprises. It is necessary to provide a detailed analysis of two case studies on ProM.

Perspectives According to the limitations of D2FD method discussed above, we present some perspectives. To extract event logs, we always focus on a single view from the source of data. If we want to change a view and generate a new event log to gain more knowledge, it is necessary to build models between process and data (De Murillas et al., 2016). Based on this model, the general approach would be "scope, bind and classify" to create a new event log, where scope determines the relevant events, bind relates events to process instances and classify relates process instances to processes (Van der Aalst, 2015); Over-fitting problem can be solved in the stage of TS model to make sure that the initial TS is general enough. We can also design a aggregation and abstraction method to merge and remove the states or activities which contain low-level detail information (Günther and Van der Aalst, 2007); Last activity problem happens because an initial state is created in the Fuzzy-DEVS model. Same as other process modeling languages, a new Fuzzy-DEVS formalism

appropriate to process modeling needs to propose two new functions which represent initial state and end state respectively; The ontology alignment can be realized through an integration model in which semantic heterogeneity provides semantic mapping of heterogeneous data in the cluster environment. The results provide references for decision-making (Zhou, 2016). A pattern-based core word algorithm (Song et al., 2013) can also help to measure the semantic similarity between a pair of compound words; We also anticipate to present a detailed analysis of case studies and make validation directly with enterprises.

Publications The essential ideas and results in this thesis have been validated by publications cited below. The first idea of transforming from event logs to DEVS is explained in C1. Then this idea is extended to get a Fuzzy-DEVS model using dependency method in C2. The D2FD method is proposed in the journal paper J1 which extends the analysis of event data and AFTC. J1 presents the method from event data to event logs and the methods from TS to Fuzzy-DEVS atomic model. Fuzzy cluster is proposed to construct a Fuzzy-DEVS coupled model in C4. A corresponding case study with the complete D2FD method is shown in C5. In C3, the validation methods associated with theory of modeling and simulation is referred and a new paradigm of inferring models and simulations is proposed. The idea of integrating Granger Causality is first discussed in C6.

J1 Wang Yan and Zacharewicz Grégory and Traoré Mamadou Kaba and Chen David. An integrative approach to simulation model discovery: Combining system theory, process mining and fuzzy logic. Journal of Intelligent & Fuzzy Systems (Impact Factor 2017: 1.261), 34(1): 477-490, 2018.

C1 Wang Yan and Zacharewicz Grégory and Traoré Mamadou Kaba and Chen David. A proposal of using DEVS model for process mining. 27th European Modeling & Simulation Symposium (Simulation in Industry). 403-409, Bergeggi, Italy, 2015.

C2 Wang Yan and Zacharewicz Grégory and Traoré Mamadou Kaba and Chen David. Integrating dependency with DEVS in the process mining. New Information Communication Sciences and Technology for Sustainable Development International France-China Workshop, Bordeaux, France, 2015.

C3 Wang Yan and Zacharewicz Grégory and Traoré Mamadou Kaba and Chen David. Verification and validation of D2FD method. New Information Communication Sciences and Technology for Sustainable Development International France-China Workshop, Clermond Ferrand, France, 2017.

C4 Wang Yan and Zacharewicz Grégory and Traoré Mamadou Kaba and Chen David. Use fuzzy clustering for discrete event simulation model construction. IFAC 2017 World Congress, The 20th World Congress of the International Federation of Automatic Control, Toulouse, France, 2017.

C5 Wang Yan and Zacharewicz Grégory and Traoré Mamadou Kaba and Chen David. A tool for mining discrete event simulation model. Winter Simulation Conference (WSC), 2017 Winter. IEEE, 3066-3077. Las Vegas, United States, 2017.

C6 Wang Yan and Zacharewicz Grégory and Traoré Mamadou Kaba and Chen David. A predictive validation method for discovering discrete event simulation models. Les journées DEVS francophones applications de la théorie de la modélisation et de la simulation (JDF). Cépaduès, 19-20. Cargèse, France, 2018.

References

Agostinho, C., J. Sarraipa, F. D'Antonio, and R. Jardim-Gonçalves (2007). Enhancing step-based interoperabity using model morphisms. In *Enterprise Interoperability II*, pp. 817–828. Springer.

Agostinho, C., J. Sarraipa, D. Goncalves, and R. Jardim-Goncalves (2011). Tuple-based semantic and structural mapping for a sustainable interoperability. In *Doctoral Conference on Computing, Electrical and Industrial Systems*, pp. 45–56. Springer.

Barros, F. J. (1995). Dynamic structure discrete event system specification: a new formalism for dynamic structure modeling and simulation. In *Proceedings of the 27th conference on Winter simulation*, pp. 781–785. IEEE Computer Society.

Barros, F. J. (1997). Modeling formalisms for dynamic structure systems. *ACM Transactions on Modeling and Computer Simulation (TOMACS) 7*(4), 501–515.

Bazoun, H., G. Zacharewicz, Y. Ducq, and H. Boye (2014). Business process simulation: Transformation of bpmn 2.0 to devs models. In *Proceedings of the Symposium on Theory of Modeling & Simulation-DEVS Integrative M&S Symposium*, pp. 13–16.

Bender, E. A. (2012). *An introduction to mathematical modeling*. Courier Corporation.

Bisgambiglia, P. A. (2008). *Approche de modélisation approximative pour des systèmes à événements discrets: Application à l'étude de propagation de feux de forêt*. Ph. D. thesis, Université Pascal Paoli.

Bisgambiglia, P. A., L. Capocchi, P. Bisgambiglia, and S. Garredu (2010). Fuzzy inference models for discrete event systems. In *Fuzzy Systems (FUZZ), 2010 IEEE International Conference on*, pp. 1–8. IEEE.

Bisgambiglia, P. A., E. De Gentili, P. Bisgambiglia, and J.-F. Santucci (2008). Discrete events system simulation-based defuzzification method. In *Electrotechnical Conference, 2008. MELECON 2008. The 14th IEEE Mediterranean*, pp. 132–138. IEEE.

Bisgambiglia, P. A., E. d. Gentili, P. Bisgambiglia, and J. F. Santucci (2008, May). Fuzzy modeling for discrete events systems. In *MELECON 2008 - The 14th IEEE Mediterranean Electrotechnical Conference*, pp. 151–157.

Bouanan, Y., M. B. El Alaoui, G. Zacharewicz, and B. Vallespir (2014). Using devs and cell-devs for modelling of information impact on individuals in social network. In *IFIP International Conference on Advances in Production Management Systems*, pp. 409–416. Springer.

Bunge, M. (2017). *Causality and modern science*. Routledge.

Cameranesi, M., C. Diamantini, L. Genga, and D. Potena (2017). Students' careers analysis: a process mining approach. In *Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics*, pp. 26. ACM.

Castro, R., E. Kofman, and G. Wainer (2008). A formal framework for stochastic devs modeling and simulation. In *Proceedings of the 2008 Spring simulation multiconference*, pp. 421–428. Society for Computer Simulation International.

Chen, Y. Y. and T. C. Tsao (1989). A description of the dynamic behavior of fuzzy systems. *IEEE Transactions on Systems, Man, and Cybernetics 19*(4), 745–755.

Cho, S. M. and T. G. Kim (1998). Real-time devs simulation: Concurrent, time-selective execution of combined rt-devs model and interactive environment. In *Proceeding of 1998 Summer Simulation Conference, Reno, Nevada*, pp. 90.

Chow, A. C. H. and B. P. Zeigler (1994). Parallel devs: A parallel, hierarchical, modular modeling formalism. In *Simulation Conference Proceedings, 1994. Winter*, pp. 716–722. IEEE.

D'Aquin, M. and A. Gangemi (2011). Is there beauty in ontologies? *Applied Ontology 6*(3), 165–175.

De Medeiros, A. K. A., A. J. Weijters, and W. M. Van der Aalst (2007). Genetic process mining: an experimental evaluation. *Data Mining and Knowledge Discovery 14*(2), 245–304.

De Murillas, E. G. L., H. A. Reijers, and W. M. Van der Aalst (2016). Connecting databases with process mining: a meta model and toolset. In *Enterprise, Business-Process and Information Systems Modeling*, pp. 231–249. Springer.

Dhamala, M., G. Rangarajan, and M. Ding (2008). Analyzing information flow in brain networks with nonparametric granger causality. *Neuroimage 41*(2), 354–362.

Diamantini, C., L. Genga, D. Potena, and W. M. Van der Aalst (2016). Building instance graphs for highly variable processes. *Expert Systems with Applications 59*, 101–118.

Dubois, D., L. Foulloy, G. Mauris, and H. Prade (2004). Probability-possibility transformations, triangular fuzzy sets, and probabilistic inequalities. *Reliable computing 10*(4), 273–297.

Dubois, D. and H. Prade (1988). Theory of possibility an approach to computerized processing of uncertainty.

Dubois, D. and H. Prade (1992). Putting rough sets and fuzzy sets together. In *Intelligent Decision Support*, pp. 203–232. Springer.

Elzas, M. S. (1984). System paradigms as reality mappings. In *Simulation and Model-Based Methodologies: An Integrative View*, pp. 41–67. Springer.

Gajski, D. D., S. Abdi, A. Gerstlauer, and G. Schirner (2009). *Embedded system design: modeling, synthesis and verification*. Springer Science & Business Media.

Geweke, J. F. (1984). Measures of conditional linear dependence and feedback between time series. *Journal of the American Statistical Association 79*(388), 907–915.

Giambiasi, N., B. Escude, and S. Ghosh (2001). Gdevs: A generalized discrete event specification for accurate modeling of dynamic systems. In *5th International Symposium on Autonomous Decentralized Systems.*, pp. 464–469. IEEE.

Giambiasi, N., M. Smaili, and C. Frydman (1994). Discrete event simulation with fuzzy times. In *European Simulation Symposium*, Turkey.

Gore, R. and S. Diallo (2013). The need for usable formal methods in verification and validation. In *Winter Simulation Conference (WSC)*, pp. 1257–1268. IEEE.

Granger, C. W. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: Journal of the Econometric Society*, 424–438.

Greco, G., A. Guzzo, L. Pontieri, and D. Sacca (2006). Discovering expressive process models by clustering log traces. *IEEE Transactions on Knowledge and Data Engineering 18*(8), 1010–1027.

Günther, C. W. and W. M. Van der Aalst (2007). Fuzzy mining–adaptive process simplification based on multi-perspective metrics. In *International Conference on Business Process Management*, pp. 328–343. Springer.

Günther, C. W. and E. Verbeek (2009). Xes standard definition. *Fluxicon Process Laboratories 13*, 14.

Hamri, M. E.-A., N. Giambiasi, and C. Frydman (2006). Min–max-devs modeling and simulation. *Simulation Modelling Practice and Theory 14*(7), 909–929.

Hand, D. J. (2007). Principles of data mining. *Drug safety 30*(7), 621–622.

Harbola, A., D. Negi, and D. Harbola (2012). Infinite automata and formal verification. *International Journal 2*(3).

Heidari, F. and N. Assy (2016). Usage analytics using process mining. In *12th International Workshop on Business Process Intelligence (BPI)*. Key Findings for the Dutch Employee Insurance Agency.

Höppner, F. (1999). *Fuzzy cluster analysis: methods for classification, data analysis and image recognition*. John Wiley & Sons.

Hu, X. (2004). A simulation-based software development methodology for distributed real-time systems.

InterOP, N. (2005). consortium, deliverable dtg3. 2: Tg momo roadmap. *Google Scholar*.

Jacques, C. J. and G. A. Wainer (2002). Using the cd++ devs toolkit to develop petri nets. In *Summer Computer Simulation Conference*, pp. 51–56. Society for Computer Simulation International; 1998.

Jalali, A. (2016). Exploring different aspects of users behaviours in the dutch autonomous administrative authority through process cubes. *Business Process Intelligence (BPI) Challenge*.

Jans, M., M. G. Alles, and M. A. Vasarhelyi (2014). A field study on the use of process mining of event logs as an analytical procedure in auditing. *The Accounting Review 89*(5), 1751–1773.

Kaufman, L. and P. J. Rousseeuw (2009). *Finding groups in data: an introduction to cluster analysis*, Volume 344. John Wiley & Sons.

Keller, R. M. (1976). Formal verification of parallel programs. *Communications of the ACM 19*(7), 371–384.

Khan, M. S. (2008). Fuzzy time control modeling of discrete event systems. *ICIAR-51, WCECS*, 683–688.

Kim, J. and B. P. Zeigler (1996). Designing fuzzy logic controllers using a multiresolutional search paradigm. *IEEE Transactions on Fuzzy Systems 4*(3), 213–226.

Kim, S., H. S. Sarjoughian, and V. Elamvazhuthi (2009). Devs-suite: a component-based simulation tool for rapid experimentation and evaluation. In *Spring Simulation Multi-conference, San Diego, CA, USA*.

Klir, G. J. (2013). *Architecture of systems problem solving*. Springer Science & Business Media.

Kočí, R. and V. Janoušek (2009). Simulation based design of control systems using devs and petri nets. In *International Conference on Computer Aided Systems Theory*, pp. 849–856. Springer.

Kutz, O. and J. Hois (2012). Modularity in ontologies.

Kwon, Y. W., H. C. Park, S. H. Jung, and T. G. Kim (1996). Fuzzy-devs formalism: concepts, realization and applications. In *Proceedings AIS*, pp. 227–234.

Law, A. M. (2008). How to build valid and credible simulation models. In *Winter Simulation Conference (WSC)*, pp. 39–47. IEEE.

Lee, C. C. (1990). Fuzzy logic in control systems: fuzzy logic controller. i. *IEEE Transactions on systems, man, and cybernetics 20*(2), 404–418.

Lee, J. and S. Chi (2005). Using symbolic devs simulation to generate optimal traffic signal timings. *Simulation 81*(2), 153–170.

Michiardi, P. and R. Molva (2002). Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In *Advanced communications and multimedia security*, pp. 107–121. Springer.

Mieke, J. (2015). From data to event log. *Process Mining Camp*.

Mitchell, M. (1998). *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press.

Oren, T. (1987). Taxonomy of simulation model processing.

Pace, D. K. (2000). Ideas about simulation conceptual model development. *Johns Hopkins APL technical digest 21*(3), 327–336.

Peterson, J. L. (1977). Petri nets. *ACM Computing Surveys (CSUR) 9*(3), 223–252.

Petri, C. A. and W. Reisig (2008). Petri net. *Scholarpedia 3*(4), 6477.

Piu, C. (2010). Simulation games: Ontology. *Simulation and Gaming for Mathematical Education: Epistemology and Teaching Strategies*, 25.

Quesnel, G., R. Duboz, and É. Ramat (2009). The virtual laboratory environment–an operational framework for multi-modelling, simulation and analysis of complex dynamical systems. *Simulation Modelling Practice and Theory 17*(4), 641–653.

Roebroeck, A., E. Formisano, and R. Goebel (2011). The identification of interacting networks in the brain using fmri: model selection, causality and deconvolution. *Neuroimage 58*(2), 296–302.

Ross, T. J. (2009). *Fuzzy logic with engineering applications*. John Wiley & Sons.

Santucci, J. F. and L. Capocchi (2014). Fuzzy discrete-event systems modeling and simulation with fuzzy control language and devs formalism. In *Sixth International Conference on Advances in System Simulation (SIMUL2014)*, pp. 250–255. Citeseer.

Sargent, R. G. (2009). Verification and validation of simulation models. In *Winter Simulation Conference (WSC)*, pp. 162–176. IEEE.

Scheer, A. W. (1998). *Business Process Engineering: Reference Models for Industrial Enterprises*. Business process engineering / August-Wilhelm Scheer. Springer.

Scheps, S. (2011). *Business intelligence for dummies*. John Wiley & Sons.

Schlesinger, S. (1979). Terminology for model credibility. *Simulation 32*(3), 103–104.

Seo, C., B. P. Zeigler, R. Coop, and D. Kim (2013). Devs modeling and simulation methodology with ms4 me software tool. In *Proceedings of the Symposium on Theory of Modeling & Simulation-DEVS Integrative M&S Symposium*, pp. 33. Society for Computer Simulation International.

Seth, A. K. (2007). Granger causality. *Scholarpedia 2*(7), 1667.

Seth, A. K. (2010a). A matlab toolbox for granger causal connectivity analysis. *Journal of neuroscience methods 186*(2), 262–273.

Seth, A. K. (2010b). Measuring autonomy and emergence via granger causality. *Artificial life 16*(2), 179–196.

Simon, H. A. (1991). The architecture of complexity. In *Facets of systems science*, pp. 457–476. Springer.

Song, F., G. Zacharewicz, and D. Chen (2013). Pattern-based core word recognition to support ontology matching. *International Journal of Knowledge-based and Intelligent Engineering Systems 17*(2), 167–176.

Song, H. S. and T. G. Kim (1994). The devs framework for discrete event systems control. In *AI, Simulation, and Planning in High Autonomy Systems. Distributed Interactive Simulation Environments., Proceedings of the Fifth Annual Conference on*, pp. 228–234. IEEE.

Ter Hofstede, A. H., W. M. Van der Aalst, M. Adams, and N. Russell (2009). *Modern Business Process Automation: YAWL and its support environment*. Springer Science & Business Media.

Thaler, T., S. Knoch, N. Krivograd, P. Fettke, and P. Loos (2014). Itil process and impact analysis at rabobank ict. *BPI Challenge*.

Tolk, A. (2012). *Ontology, epistemology, and teleology for modeling and simulation: Philosophical foundations for intelligent M&S applications*, Volume 44. Springer.

Traoré, M. K. (2008). A next generation modeling and simulation framework. In *Proceedings of the International Conference on Simulation Tools and Techniques for Communications, Networks and Systems*.

Uhrmacher, A. M. (2001). Dynamic structures in modeling and simulation: a reflective approach. *ACM Transactions on Modeling and Computer Simulation (TOMACS) 11*(2), 206–232.

Van der Aalst, W. M. (1998). The application of petri nets to workflow management. *Journal of circuits, systems, and computers 8*(01), 21–66.

Van der Aalst, W. M. (2011). *Process Mining - Discovery, Conformance and Enhancement of Business Processes*. Springer.

Van der Aalst, W. M. (2015). Extracting event data from databases to unleash process mining. In *BPM-Driving innovation in a digital world*, pp. 105–128. Springer.

Van der Aalst, W. M. (2016). *Process mining: data science in action*. Springer.

Van der Aalst, W. M., A. Adriansyah, and B. Van Dongen (2011). Causal nets: a modeling language tailored towards process discovery. In *International conference on concurrency theory*, pp. 28–42. Springer.

Van der Aalst, W. M., T. Weijters, and L. Maruster (2004). Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering 16*(9), 1128–1142.

Van Tendeloo, Y. (2013). Research internship i: Efficient devs simulation.

Van Tendeloo, Y. and H. Vangheluwe (2014). The modular architecture of the python (p) devs simulation kernel: work in progress paper. In *Proceedings of the Symposium on Theory of Modeling & Simulation-DEVS Integrative*, pp. 14. Society for Computer Simulation International.

Wagner, F. (2005). Moore or mealy model. *States works, Technical notes http://stateworks. com*.

Wainer, G. and N. Giambiasi (2001). Timed cell-devs: modeling and simulation of cell spaces. In *Discrete event modeling and simulation technologies*, pp. 187–214. Springer.

Weijters, A. and J. Ribeiro (2011). Flexible heuristics miner (fhm). In *Computational Intelligence and Data Mining (CIDM), 2011 IEEE Symposium on*, pp. 310–317. IEEE.

Weske, M. (2012). Business process management architectures. In *Business Process Management*, pp. 333–371. Springer.

Willems, J. C. (1989). Models for dynamics. In *Dynamics reported*, pp. 171–269. Springer.

Youcef, D. and H. Maamar (2014). Specification of the state lifetime in the devs formalism by fuzzy controller. *arXiv preprint arXiv:1401.5638*.

Zacharewicz, G. (2006). *Un environnement G-DEVS/HLA: Application à la modélisation et simulation distribuée de workflow*. Ph. D. thesis, Université de droit, d'économie et des sciences-Aix-Marseille III.

Zacharewicz, G., C. Frydman, and N. Giambiasi (2008). G-devs/hla environment for distributed simulations of workflows. *Simulation 84*(5), 197–213.

Zadeh, L. A. (1996). Fuzzy sets. In *Fuzzy Sets, Fuzzy Logic, And Fuzzy Systems: Selected Papers by Lotfi A Zadeh*, pp. 394–432. World Scientific.

Zeigler, B. P., G. Ball, H. Cho, J. Lee, and H. Sarjoughian (1999). Implementation of the devs formalism over the hla/rti: Problems and solutions. In *Simulation Interoperation Workshop (SIW)*, Volume 99.

Zeigler, B. P. and P. E. Hammonds (2007). *Modeling & simulation-based data engineering: introducing pragmatics into ontologies for net-centric information exchange*. Elsevier.

Zeigler, B. P., Y. Moon, V. L. Lopes, and J. Kim (1996). Devs approximation of infiltration using genetic algorithm optimization of a fuzzy system. *Mathematical and Computer Modelling 23*(11-12), 215–228.

Zeigler, B. P., H. Praehofer, and T. G. Kim (2000). *Theory of modeling and simulation: integrating discrete event and continuous complex dynamic systems*. Academic press.

Zeigler, B. P. and H. S. Sarjoughian (2003). Introduction to devs modeling and simulation with java: Developing component-based simulation models. *Technical Document, University of Arizona*.

Zeigler, B. P. and H. S. Sarjoughian (2013). Devs integrated development environments. In *Guide to Modeling and Simulation of Systems of Systems*, pp. 11–26. Springer.

Zhou, Q. (2016). Research on heterogeneous data integration model of group enterprise based on cluster computing. *Cluster Computing 19*(3), 1275–1282.

Zimmermann, H. J. (1996). Fuzzy control. In *Fuzzy Set Theoryand Its Applications*, pp. 203–240. Springer.