

ULRR

The use of formal methods to describe a financial crime analytics model

Item Type	Thesis
Authors	Nagle, Daniel
Download date	2026-05-17 00:17:01
Item License	https://creativecommons.org/licenses/by-nc-sa/1.0/
Link to Item	https://hdl.handle.net/10344/8172

The use of formal methods to describe a financial crime analytics model



UNIVERSITY of LIMERICK
OLLSCOIL LUIMNIGH

Daniel Nagle

Computer Science & Information Systems

Faculty of Science & Engineering

University of Limerick

Submitted to the University of Limerick for the degree of

Master of Science

2019

1. Supervisor: Dr. Patrick Healy
Computer Science & Information Systems
University *of* Limerick
Ireland

Abstract

There are many examples of the utilization of formal methods for creating complete specifications of software applications. Often these are bounded in scope to the concept of a critical system. This work looks to expand the definition of *critical systems* to include systems that provide safety measures outside of contexts like utilities or military. The expanded definition is an exploration of the concept of financial crime detection measures and their specifications, a vital element in counter terrorism controls that highlight the movement of terrorists and their funding mechanisms.

Within the context of financial crime detection systems we explore if formal methods can be used in the description of software from this industrial application. The resulting analysis of this research is the selection of a formal method for the documentation of a specific example of a financial crime model from industry. We further analyze the applicability of the selected method for industrial software development and survey industry experts to understand if formal methods has a valid place in the software development life cycle for financial crime solutions.

Declaration

I herewith declare that I have produced this thesis without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such. This paper has not previously been presented in identical or similar form to any other Irish or foreign examination board.

The thesis work was conducted under the supervision of Dr. Patrick Healy at the University of Limerick.

Limerick, 2019

Acknowledgements

I would like to thank Dr. Patrick Healy, Dr. Michael English and Prof. David L. Parnas for their guidance and help throughout my Masters degree. Without their guidance and comments much of the work would not have evolved to reach it's final form. In addition, I wish to thank Kim Nagle, Colm Quinn, Christopher Long, Brian Matschke, Kevin Luh, David Stewart, Darren Bane, Marius Dragomiroiu and SAS for their guidance, examples, reviews and commentary of my work.

I am exceedingly grateful for the privilege of working on interesting new research, contributing to a new use case for utilizing tabular expressions and applying the Trace Function Method to broaden the purview of my work. I wish to acknowledge the financial support of the Science Foundation Ireland under SFI grant 01/P1.2/C009 for part of this research.

Finally, I would like to thank my friends and family for their support and patience. I would also like to give a special thanks to my daughters Alanah and Aisling for all the joy they brought me in finalizing this work and helping me be better at managing my priorities.

This work is dedicated to my Grandfather, Daniel; Mother, Eileen; and my wife, Kim. The three people that always make me want to be a better person.

Contents

List of Tables	ix
List of Figures	xi
1. Introduction	1
1.1. Background	1
1.2. Motivation	4
1.3. Related Work	5
1.4. Thesis Outline	6
2. Literature Review: Formal Methods	9
2.1. Overview	9
2.2. Introduction	9
2.3. Formal Method Description Types	10
2.3.1. Parnas 1972 Method	11
2.3.2. Z	11
2.3.3. VDM	12
2.3.4. B	12
2.3.5. Trace Assertion Method	12
2.3.6. Trace Function Method	13
2.4. Understanding Tabular Notation Past and Present	14
2.5. Parnas Predicate Logic	18
2.5.0.1. Example Tabular Expression Tree	19
2.6. Applying the TFM	23
2.6.1. Tooling Supporting Tabular Formal Methods	26
2.7. Summary	32
3. Use Case Legalities	35
3.1. Introduction	35
3.2. FATF	36
3.3. The Fourth EU Directive on Anti-Money Laundering	39
3.4. Conclusion	41

4. Financial Crime Detection Technologies	43
4.1. Overview	43
4.2. Financial Crime Detection as a Critical System	43
4.3. Financial Security System Specifications	45
4.3.1. Detection Techniques	51
4.3.1.1. Link Analysis	57
4.3.1.2. Text Analytics	61
4.3.1.3. Neural Networks	64
4.3.1.4. Bayesian Networks	67
4.3.1.5. Positive and Negative Match	69
4.4. Financial Crime Model Examples	71
4.4.1. Name Recognition in Financial Crime Applications	72
4.4.2. A Neural Networking Fraud Detection Example	74
4.5. Analysis	79
4.6. Summary	82
5. Use Case and Code Review	83
5.1. Overview	83
5.2. Customer Due Diligence	83
5.3. Formal Decomposition	84
5.3.1. CDD Model Overview	90
5.3.2. Top Level Aggregation Rules	91
5.3.3. Compliance Element Rule Groups	94
5.3.4. Rule Group CDDRG_C010	94
5.3.4.1. CDDR_C101	94
5.3.5. Rule Group CDDRG_C020	96
5.3.5.1. CDDR_C910	96
5.3.6. Rule Group CDDRG_C030	96
5.3.6.1. CDDR_C920	96
5.3.7. Rule Group CDDRG_E010	97
5.3.7.1. CDDR_E101	98
5.3.8. Rule Group CDDRG_P010	99
5.3.8.1. CDDR_P010	99

5.3.9.	Rule Group CDDRG_X010	100
5.3.9.1.	CDDR_X010	100
5.3.10.	Rule Group CDDRG_X015	101
5.3.10.1.	CDDR_X015	102
5.3.11.	Rule Group CDDRG_X020	103
5.3.11.1.	CDDR_X020	103
5.3.11.2.	CDDR_X030	104
5.4.	TFM	105
5.4.1.	Using the TFM	106
5.4.2.	Component Programs and Inputs	108
5.4.3.	Set Elements	111
5.4.4.	Auxiliary Functions	113
5.4.5.	Output Messages	114
5.4.6.	Output Function	115
5.4.6.1.	Country Category Output Functions	116
5.4.6.2.	Entity Category Output Functions	118
5.4.6.3.	Product Category Output Functions	120
5.4.6.4.	Final Program Output Function	121
5.4.7.	Reading the TFM	129
5.5.	TFM Analysis	131
5.6.	Summary	135
6.	Testing & Discussion	137
6.1.	Introduction	137
6.2.	Coverage Testing	138
6.2.1.	Testing Strategies	141
6.2.2.	Analysis	143
6.3.	Comprehensibility Testing	143
6.3.1.	Survey Structure	144
6.3.2.	Results	145
6.4.	Summary	148

7. Conclusion	151
7.1. Results	151
7.1.1. Review of Formal Methods Technology and Introduction to the Trace Function Method	151
7.1.2. Model Legal Reference Analysis	152
7.1.3. An analysis of financial crime detection systems, with an in-depth review of how they attempt to systematically solve for financial crime activity	153
7.1.4. Trace Function Method Description of a Financial Crime Model	153
7.1.5. Use Case Conclusions	154
7.1.6. Survey into Industrial application	155
7.2. Critical Analysis	156
7.3. Future Work	158
7.4. Summary	159
8. Appendix 1	161
8.1. CDD Use Case SAS Code	161
8.1.0.1. CDDR_C101	161
8.1.0.2. CDDR_C102	162
8.1.0.3. CDDR_C103	163
8.1.0.4. CDDR_C104	164
8.1.1. Rule Group CDDRG_C020	164
8.1.1.1. CDDR_C910	165
8.1.2. Rule Group CDDRG_C030	165
8.1.2.1. CDDR_C920	165
8.1.2.2. CDDR_E101	166
8.1.2.3. CDDR_E102	167
8.1.2.4. CDDR_E103	168
8.1.2.5. CDDR_E104	168
8.1.2.6. CDDR_E105	169
8.1.2.7. CDDR_E106	170
8.1.2.8. CDDR_E107	171
8.1.2.9. CDDR_E108	172

8.1.2.10.	CDDR_E109	173
8.1.2.11.	CDDR_E110	174
8.1.2.12.	CDDR_E201	175
8.1.2.13.	CDDR_E202	176
8.1.2.14.	CDDR_E203	177
8.1.2.15.	CDDR_E204	178
8.1.3.	Rule Group CDDRG_E020	179
8.1.3.1.	CDDR_E910	179
8.1.4.	Rule Group CDDRG_E030	179
8.1.4.1.	CDDR_E920	179
8.1.4.2.	CDDR_P010	180
8.1.5.	Rule Group CDDRG_P020	180
8.1.5.1.	CDDR_P910	180
8.1.6.	Rule Group CDDRG_P030	181
8.1.6.1.	CDDR_P920	181
8.1.6.2.	CDDR_X010	181
8.1.6.3.	CDDR_X015	182
8.1.6.4.	CDDR_X020	183
8.1.6.5.	CDDR_X030	183
9.	Appendix 2	185
9.1.	A Survey on understanding software specifications written in formal methods	185
9.1.1.	Instructions	185
9.1.2.	Introduction	185
9.1.3.	Date Management Software Example	186
9.1.4.	Time Management Software Example	189
9.1.5.	Stack Example	190
9.1.6.	A CDD Program Scoring Banking Products	191
9.1.6.1.	CDDR_P010	191
9.1.6.2.	CDDR_P010	191
9.1.6.3.	CDDR_P910	192
9.1.6.4.	CDDR_P920	192

9.1.7. Comprehension Questions	195
9.1.8. Regulatory Questions	196
9.1.9. Utility Questions	197
Bibliography	199

List of Tables

2.1. Normal Function Table	21
5.1. Country Rule Element Aggregation	92
5.2. Entity Rule Element Aggregation	93
5.3. Product Rule Element Aggregation	93
5.4. Final Score Rule Element Aggregation	94
5.5. C101 Parameters	95
5.6. E101 Parameters	98
5.7. P010 Parameters	99
5.8. X010 Parameters	100
5.9. X015 Parameters	102
5.10. X020 Parameters	104
5.11. X030Parameters	105
5.12. PGM(FinalScore)	108
5.13. PGM(CountryScore)	109
5.14. PGM(EntityScore)	109
5.15. PGM(ProductScore)	110
5.16. PGM(AutoHigh)	110
5.17. Elements of Set CountryScore	111
5.18. Elements of Set EntityScore	112
5.19. Elements of Set ProductScore	112
5.20. CountryScore Auxiliary Functions	113
5.21. EntityScore Auxiliary Functions	114
5.22. ProductScore Auxiliary Functions	114
5.23. Set of Output Messages	115
5.24. Output Function Country Group Rule Score Function	117
5.25. Country Max Attribute Value	118
5.26. Country Category Aggregate Score Logic	119
5.27. Output Function Entity Group Rule Score Function Template	122
5.28. Entity Max Attribute Value	123
5.29. Entity Category Aggregate Score Logic	124

5.30. Output Function Product Group Rule Score Function Template	125
5.31. Product Max Attribute Value	126
5.32. Product Category Aggregate Score Logic	127
5.33. Final Output Function	128
8.1. C102 Parameters	162
8.2. C103 Parameters	163
8.3. C104 Parameters	164
8.4. E102 Parameters	167
8.5. E103 Parameters	168
8.6. E104 Parameters	169
8.7. E105 Parameters	170
8.8. E106 Parameters	171
8.9. E107 Parameters	172
8.10. E108 Parameters	172
8.11. E109 Parameters	174
8.12. E110 Parameters	174
8.13. E201 Parameters	175
8.14. E202 Parameters	176
8.15. E203 Parameters	177
8.16. E204 Parameters	178
9.1. P010 Parameters	191
9.2. PGM(ProductScore)	193
9.3. Elements of Set ProductScore	193
9.4. ProductScore Auxiliary Functions	193

List of Figures

4.1. Name Recognition Example	73
4.2. Name Recognition SOUNDEX example	74
4.3. Neural Networks Example Step 1	75
4.4. Neural Networks Example Step 2	76
4.5. Neural Networks Example Step 3	77
4.6. Neural Networks Example Step 4	77
4.7. Neural Networks Example Step 5	77
4.8. Neural Networks Example Step 6	78
5.1. Composite Risk Score	87
5.2. Country Score Category	88
5.3. Entity Score Category	89
5.4. Product Score Category	89

"Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live."

Martin Golding



Introduction

1.1 Background

Formal methods have been used as a mechanism to both validate and invalidate all forms of specifications related to the development of critical systems (Sohn and Seong (2000)). A necessary element to this process is the aspect of verification. Verification, be it manual or automated in nature, provides an independent capability to measure the accuracy of a software programs intended design versus its respective output.

There are many examples of formal methods being utilized to verify the specifications of different software examples. Often these examples are derived from software designed to be a *critical system*. A critical system is defined as being any piece of software acting as a control mechanism where through a fault in design or realization of the program results in the loss of life or causes physical harm.

This thesis addresses this concern through the examination of an example of some software from a non-traditional critical system. This research examines financial crime systems which are used by regulators and banks to detect potential money laundering, terrorist financing and human trafficking. We examine the area of *Know Your Customer* (KYC) which is an initiative recognized by law and operationalized by the financial services sector.

Financial crime control systems are a modern mechanism of determining criminal intent with regard to money laundering, terrorism, fraud and cyber-crime related

threats. Each have substantive impacts on the global population's daily activities but act as a *behind the scenes* mechanism to serve and protect. Through the application of a case study, we attempt to generate a formal representation of the case study application model in order to assess the applicability of the formal method approach in describing software from this industry. Further, this work examines some strategies to test the completeness of the formal representations and surveys industry experts to understand how easily comprehended these representations are. We examine the technologies used in financial crime to provide the reader with a comprehensive understanding of the techniques used to ground our case study through a survey.

Formal method representations of software are aimed at software developers wishing to improve their software quality. While there are many types of formal methods this work focuses on original work in the areas of tabular expressions, such as Group (1997). Tabular expressions are a method of representing key information pertaining to the inputs and outputs in an easy-to-read table that developers can be trained to understand Janicki et al. (1996). The advantages of such tables are their comprehensibility and the ability of a user to verify the accuracy of their software specification using an automated or manual verification system. The use of tabular expressions has proven to be invaluable in past experiments, such as Heninger et al. (1978) or Parnas et al. (1987).

To most developers, formal methods represent an arduous adjustment to the creative process of code development given the lack of industrial tooling and education to support its use. However, to ensure the safety and accuracy of software intended to support critical systems, formal methods represent a valuable means of examination and validation that this thesis argues should be explored for financial crime use cases.

The goal of the tabular representations is to allow software developers to develop software with documentation that explicitly shows how the software operates under all conditions the system could experience. Previous work in this area by Quinn (2007) explored an industrial example for telecommunications examining the validity of the approach. This work further builds on that existing research and focuses on the evaluation of an industrial example from a very different domain to investigate whether the

use formal methods in that specific context adds value by improving alignment to government regulations and improves on current methods employed in designing software for this purpose.

Further, the use of formal methods allows for a verification process such as Jing (2000) to be employed and thereby highlights to the user whether or not their software code will accurately map to the specification and satisfy all restrictions per the design. Once verified, the software developer should have higher confidence to argue the completeness of the formal representation of the software. The availability of appropriate tooling, such as verification capabilities, is critical in order to enable developers to take advantage of a specific formal method. The absence of tooling being present in industrial form makes the use of a formal method in industry unlikely.

As developers would have a detailed set of proven mathematical representations of the code base in this scenario, expansions of the code base and documentation could be maintained accurately, ensuring the integrity of mappings to legislation. As such, this methodology is intended to provide an ongoing mechanism to ensure the clarity, longevity and accuracy of a code base and its associated documentation. The software specification could thus be leveraged by domain experts to ensure that the design rational makes sense for the domain problem and its attributes being solved.

The goal of this research is to build on this background, to survey popular formal method approaches to drafting specification documents of software programs through a literature review. Further, it will provide an argument for selecting one of these methods and explain its use through examples and will explore the availability of supporting tooling. It will introduce the topic of financial crime detection and how it is currently regulated and survey the technologies used to detect financial irregularities. Next, we attempt to answer if formal methods can successfully specify an industry financial crime model. Finally, we attempt to discover if we can adequately find strategies to solve for coverage and how comprehensible the specification is to industry experts.

1.2 Motivation

This research examines critical systems from a different perspective and attempts to build a foundation for improving our financial control systems. The motivation behind this is driven by the impact that factors such as terrorism, criminal activity and corrupt practices has on global financial systems. By applying formal methods to financial crime detection system documentation, we should be able to enforce a stronger defence against malicious behaviours by both understanding software better and ensuring that unintentional security gaps are not present in these systems.

We argue that formal methods is one approach that could improve the confidence of banks and regulators in software deployed in this industry. To validate that assertion it is necessary to have both a formal documentation standard and a mechanism for validating it. Hence the necessity for building on the existing work of Jing (2000) and Quinn (2007) to query the usefulness of formal methods in an environment that deals exclusively with time-sensitive, large-volume, complex data. By proving the relevancy of formal methods and specifically tabular expressions in a financial ecosystem, it is hoped that we can create an adaptive means to encase financial software systems with rigour and undermine criminal efforts to take advantage of such systems.

The motivation to carry out research in this area is to ensure that software engineers can align with design specifications and improve the overall quality of their software by using a formal method in their software design. This case study presents valuable insight into how we try and validate criminal activity in different contexts. As the scope of criminal intent touches almost every aspect of our lives it is important to validate that the right governance is applied to the software that protects our finances. As such, creating a specification means that can be used to generate accurate representations of these systems has the potential to alleviate some of the risk associated with the implementation of these capabilities.

Ultimately, it is our intention that the reader be able to understand by the conclusion of this work if formal methods has a place in the development of software for financial crime applications. Second, if there is a method that offers more utility than others.

Thirdly, to understand via a literature survey what technologies are employed by financial crime detection systems today. Further, if said method offers a mechanism to test its accuracy either through manual or automated means and the availability of tooling to support this endeavour. Finally, to determine if readers of specifications written in the selected formal method truly can understand them and can see utility in using them in their work.

1.3 Related Work

Previous work carried out on tabular representation based formal methods includes research by Quinn (2007) and further by Group (1997), Kahl (2003), Heitmeyer (2001), Lawford et al. (2004) and Eles and Lawford (2011). Also related to this is work in broader formal methods-based research discussed later in this work.

With regard to research into financial crime, some work has been completed by DJ Hand Bolton and Hand (2002) regarding mechanisms to detect behaviours indicative of criminal intent. A number of these methods utilize statistical methodologies surrounding Bayesian causality and Benford clustering to show harmonization of events into clusters and further, make predictions based on past observed behaviours Bolton and Hand (2001).

Other work by Phua et al. (2005) surveys some of the technologies available and in use by vendors today to solve financial crime related problems. The survey includes several areas important to the detection of irregular financial transactions and provides a good foundation for understanding the variance of technology in this domain.

The scope of this work is to build on the utilization of these methods for the purposes of our use case and examine practical methods of the application tabular expressions to real-world criminal activity detection. However, the act of translating any detection model using formal methods has not been attempted according to our research.

1.4 Thesis Outline

The scope of this thesis is to discuss concepts regarding formal methods, software design and documentation, and research into specific software engineering problems pertinent to describing financial crime models. It is the intent of this work to examine the potential of describing financial crime detection systems with formal methods and to discover if a specific method has sufficient coverage to describe the unique aspects of this area.

Chapter 2 introduces the reader to a survey of several formal method approaches and looks at where they have been used to describe industrial examples in the past. It builds on this survey to conduct an analysis into what method would be most appropriate for this industrial application and provides further context, using examples, into how this method is applied.

Chapter 3 introduces the case study and discusses the background legislation and regulations evolving from FATF(Financial Action Task Force) and the European Union that lead the necessity of operable technological solutions to solve for financial crime detection. These regulations are what drive innovation within the financial crime space and ultimately are the driving force behind this research to ensure an easier method of ensuring compliance.

Chapter 4 is a survey of the available literature on financial crime detection technologies and illustrates the breadth of research conducted into this industrial application. It delves into several examples to showcase how formal methods could provide lift in aiding code comprehension and verification efforts. Further, it conducts an analysis on what technologies can be translated into a chosen formal method approach.

Chapter 5 examines a financial crime use case around customer due diligence and highlights code developed to solve for this concern. It illustrates the use of translating the model using the trace function method and reduces the code down to a compact formal description that encompasses all aspects of the regulatory drivers.

Chapter 6 looks at some potential coverage testing strategies that could be applied to verify the coverage of the representation. It further surveys a group of industry professionals in order to understand the applicability of formal methods in the financial crime software development sector. Further, it attempts to understand how readable a formal method specification is to these industry professionals.

Chapter 7 concludes this thesis with a discussion of what was learned from this project, its limitations and successes. It discusses what we feel are the core contributions. It also presents a breakdown of ideas that could be built upon from a foundational perspective and where technological advances could assist in improving the tool in the future. It also critically analyses the ease of use of the method and ultimately if the approach offers the utility desired for developers in this space.

"I love deadlines. I like the whooshing sound they make as they fly by."

Douglas Adams

2

Literature Review: Formal Methods

2.1 Overview

This chapter introduces the concept behind choosing a formal method to represent the software use case. It explores what work has been conducted in the past in this area. It continues by surveying formal methods existing currently that could potentially be utilized.

We explore the basis to the mathematics behind tabular expression representations and provide the reader with example expression trees to introduce them to key concepts. We discuss efforts made into the provision of tooling to support developers. We conclude by providing key elements associated with the Trace Function Method (TFM).

2.2 Introduction

This research is based around the concept of tabular expression-based documentation. Tabular expressions are a formal method approach to define scope, requirements and to act as a documentation standard for software functions. What follows is an overview of the previous research initiatives into the field. Further, commentary on the state of tooling is provided in order to establish efforts to make the methods more

main-stream for software developers. In the absence of tooling it would be very difficult to assume any form of uptake for any formal method. Industrial processes to generate software try and attempt to take the path of least resistance even for critical system design. The inference being that if it hard to create a specific representation of software for a software engineer during their day-to-day work it is highly unlikely that the method will gain ground within that industry due to the cost it would require in terms of time.

2.3 Formal Method Description Types

There exists an abundance of formal description approaches, each of which have relative strengths and weaknesses. Building on the survey of Quinn (2007) in the area we examine some of the more actively utilized methods and from there attempt to rationalize the appropriate method for use in describing this case study.

As Quinn points out, algebraic and axiomatic description types described in work such as Ehrig and Mahr (1980) and Tucker and Zucker (2002) have waned in terms of research popularity mostly due to issues in terms of the ability to express some key mathematical concepts like a finite axiomatisation. Nonetheless they formed the basis for future work and thus should not be discounted in terms of their importance to the field. Specifically, they laid a foundation for Bartussek and Parnas (1978) to build the concept of traces and trace functions upon.

Quinn also points out the utilization of abstract models by standards organizations such as Z ISO (2002) but his observation regarding uptake on these descriptions remains true today as there has been limited utilization of these standards in industry beyond academic examples. In part the reason for this lack of uptake has been the lack of available tooling and education aimed at developers regarding the use of these methods. Thus this work examines some of the other pragmatic methods that have been employed to describe an industrial example of a critical system beyond traditional algebraic and axiomatic approaches.

2.3.1 Parnas 1972 Method

This pragmatic model method originally published by Parnas (1972*b*) has seen several industrial applications primarily in the area of military research, such as Heitmeyer (2001). The basis of this method is the formal definition of a module; a software unit of work with its own discrete series of inputs and outputs. The method had the benefit of allowing the developers to remain agnostic in terms of the specific coding language utilized and reduced the burden on testing by focusing on the specifics of the module statements rather than potential interpretations. As discussed by Quinn (2007) the method has a severe drawback in the form of temporal displacement of historical values meaning only the most recent value could be referred to at any one time by the specification of the module thereby allowing the potential for errors to creep into the specification format.

2.3.2 Z

Z is an example of an abstract model description method defined originally in Bowen (1992). It utilizes mathematical abstractions to create a representation of a software system. As Quinn (2007) points out it is not possible to create an implementation of the software directly from the mathematical representation of its specification without the use of Floyd-Hoare Logic Floyd (1967). The reasoning behind this is that Z is effectively a notation-based description language that uses first order logic and set theory to describe software systems.

Rather than focus on Parnas style modules, Z uses *schema* as the atomic unit of specification and describes a component by examining the interactions between schemas. Thus, it is not possible to represent the Parnas (1972*b*) definition of a *module* in Z without further mathematical manipulation due to the inherent differences between the approaches.

Whilst Z is relatively well used due to its ISO 2002 standardization, it does suffer from the fact it cannot easily be translated into development languages for the realization of the design. Despite this inhibition the method has gained adoption by some

developers due to the availability of tool suites to aid in its utilization as a documentation standard during the software development life cycle.

2.3.3 VDM

VDM is another example of an abstract model description method that focuses on the development method. It's made up of two subcomponents; VDM Jones (1990) and VDM-SL Dawes (1991). The VDM element focuses on the abstract design descriptions and generation mechanisms, while VDM-SL is the structured description language. Due to similar limitations to Z, VDM is not actively utilized beyond academic circles. The inability to translate VDM specifications into an actual software representation has led to limited uptake in the development world.

2.3.4 B

B is an example of a collection of mathematically-based techniques originally based on both Z and VDM research, but Abrial builds upon that foundation to develop more of a fully fledged independent method Abrial (1996). *B* uses keywords as an atomic description unit and each keyword combines to form modules. A further concept used in *B* is the idea of interdependent abstract machines which are effectively state descriptors for the module.

With the exception of the Parnas-based approaches *B* has seen the broadest adoption in terms of industrial use and case studies such as the Paris metro Dehbonei and Mejia (1994). *B* has a broad range of tools and adapters that allow it to be easily translated from specification to code including several compilers. However, outside of the case studies above it has yet to be utilized more broadly probably as a result of its lack of standardization.

2.3.5 Trace Assertion Method

Bartussek and Parnas (1978) provide a definition of a trace that defined a trace as being "a description of a sequence of calls on functions with the module in the initial state". Further, it elaborates on the definition to include subtraces and the parsing of

such traces and the removal of ambiguity by actively parsing from left to right to show execution order. As such the trace assertion method is a means of describing modules through the mechanism of traces.

Quinn (2007) elaborates on the advantages and disadvantages of the method where it is pointed out that traces are a compact means of module description that can easily be extended. The logical flow of the trace means for ease of translation into a development language of choice without sacrificing the accuracy of the specification. However, while the method does create more compact descriptions in comparison to other methods it can have excessively detailed traces that can lead to complexity and difficult-to-consume information.

2.3.6 Trace Function Method

As per Quinn (2007), Parnas (2003a), Bane et al. (2004) and Parnas and Dragomiroiu (2006) the trace function method is a formal method of description that utilizes traces with other techniques in order to accurately describe software components. Applying logic to the evolution of changes of state to a component Parnas introduces the concept of events, which are discrete point in times perspectives on the internal state of the module, its variables and changes-to values. With events a trace now becomes a sequence of event descriptors which is analogous to a historic and current snapshot of the module. It is feasible then, using this method, to accurately describe inputs, outputs and the related actions occurring within the module over time.

Quinn (2007) notes several advantages of the technique. Primarily, the advantage of being able to aggregate several documents into one unified description eases the burden on developers regarding documentation overhead. This document should provide all necessary clarity with regard to inputs, outputs, module interfaces and transformations within the software. There are also advantages with regard to testing as the method lends itself towards ease of verification and validation of the descriptions. Some evaluations of the method have been observed with specific utilization in industry use cases like research conducted in Feng et al. (2011), Parnas and Vilkomir (2007) and Liu

et al. (2010). These build upon the telecommunications industry use case introduced by Quinn (2007).

2.4 Understanding Tabular Notation Past and Present

Tabular notation is a method of data representation that involves the use of expression trees. We begin to see the advantage of representing data by using this format in representing computer software using tabular notation in Janicki et al. (1996). It is immediately apparent that by using tabular notation we can document all conditions that directly impact a function and instil mathematically-driven certainty on the outcomes of a function's expected outputs. Tabular notation is stated to be a clear and concise notation and is something that anyone can be trained to read Janicki et al. (1996). Yet, although it is claimed to be easy to read¹ by involving mathematics, it, more importantly, allows us to verify aspects of software and ensure that they meet with their specification. Tabular notation has much to offer software engineers due to the diverse spectrum of information to which it can be successfully applied to without introducing ambiguities detrimental to the software's design. Tabular representations provide a means in document driven design to validate and alleviate concern over unexpected outcomes, something that has been proven to be of particular value in critical systems Parnas et al. (1987).

It is important to note for any of the variants of tabular notation that they aim to provide a set of reference documents. These reference documents are supposed to be complete representations of the software and describe the changes that occur as inputs are detected and transition through the software to create output. As such, elements such as modules, which break down the software into specific work tasks and interfaces are vital to understand the changes in state that occur as the program completes its actions. Further, these elements are important as they provide us with an understand of where error states could potentially creep in within the design. These reference documents should be mathematically based to allow for verification and easily manipulated to allow for software maintenance.

¹By individuals with a mathematical background whom have undergone some training

To fully comprehend the notation of this method of data representation we first have to understand the logic operating behind it. In Parnas (1993a) we are introduced to a very unique logic proposed by Parnas which is aimed specifically at software engineers to help them with problems in their domain. Using this logic as a mathematical foundation Janicki et al. (1996) introduces us to a notation to help us describe functions using tabular notation. Further work has continued on this subject resulting in Bane et al. (2004) which is an improved mathematical model for tabular expressions which shall be discussed later in this chapter. This work proved pivotal in the creation of the Trace Function Method (TFM) Parnas and Dragomiroiu (2006) used in our industrial example. Parnas theorized that with a suite of utilities to allow for ease of manipulation, validation and construction the use of tabular expressions for the purpose of software documentation would have the potential to enter the mainstream Parnas et al. (1994).

A verification tool for tabular expression trees is most necessary in order to provide a method of checking that a tabular expression satisfies its restriction¹ term Parnas (2003b). A human being could do this but the process would soon become tedious and as Parnas explored could become confused depending on the complexity of a given function. As such, seeking a tool that provides functionality to help us solve this problem in an automated fashion without sacrificing the ability for human beings to direct interact and adjust expression trees would provide significant assistance to developers utilizing the approach.

Since the most recent development of a tool to help 'verify' tables using third party tools we believe that there has been a change in the state of the art in the technologies surrounding mechanized mathematical tools Jing (2000), Lawford et al. (2004), Eles and Lawford (2011), Heitmeyer et al. (2005). A specialized tool could be developed for the purpose of solving our verification problem but it is not within the scope of this research.

¹A restriction is a mathematical method of representing constraints on the tabular expression

Jing's work determined that the shortest path to success was the utilization of the Prototype Verification System (PVS) Shankar et al. (1993) which has been successfully utilized for the purpose of checking the expression trees as per Rushby and Srivas (1993). While this successfully proves that PVS should be a candidate for reuse in another mechanism for validating tabular expressions it would be remiss not to evaluate the art of the possible in terms of validation/verification technologies now available. Later in this chapter, 2.6.1, we examine some of the research available into tooling for this area.

The term mechanized mathematical tools (MMTs) encompasses a large number of different types of software such as: theorem proving systems, computer algebra systems, model checkers and hybrid systems; all of which should be explored in terms of fit for purpose in tabular verification. As we are looking for a means of consuming precise documentation standards, as per Parnas et al. (1994), it is of the utmost importance that whatever candidate category of tools, and the tool itself, be able to be further validated to ensure a proven platform. Without such, the validity of verifications of all representations produced could be called into question. Rather, should these methods be adopted into industry a potential extension of this work could be to provide tooling to support developers efforts to adopt the method to their daily work.

The TFM brings some ancillary capabilities beyond that of its predecessors that are useful for those tasked with describing complex software. It is not equational or axiomatic but still allows for the use of multi-dimensional tabular expressions. Its structure allows for the limitations of the previous methods to be addressed by providing further flexibility. For example the TFM allows the documenter the ability to document modules that utilize global variables. It further simplifies important details rather than overcomplicating them while allowing for some degree of abstraction of the internal workings of the software implementation.

The core concept of the TFM is that any software module has both a global data structure and a hidden one that masks the process by which the module achieves its work. As Parnas (2006) notes, global variables are a fundamental way that modules communicate. Thus it is important to capture through our reference documentation the

intricacies of these variables and how they are transformed by a module. As we noted earlier, the concepts of *events* is another cornerstone of the TFM where the software module can be imagined as a finite state machine operating over a logical time-line delineated by important milestones. In this case the milestones are what the TFM calls *events*, and quite simply an event is where a module performs an action on either its external or hidden data structure.

Every event is different and thus should have a unique description of what has occurred. The TFM utilizes *event descriptors* for this purpose. These *event descriptors* are ultimately unique identifiers that shows the pre and post values of a specific variable prior to and post the work conducted by a software module. Finally there is a *trace*, a formal methods concept, which is simply a sequence of events that show what occurs once the module initializes.

The TFM brings all of these component elements together into a component interface document. This should have the descriptions necessary for a reader to understand the complete listing of all inputs and outputs of a given module, inclusive of the types involved. Further, it should detail the set of relations that occur tying the history of events that occurs between input and output through the form of a trace. As per Parnas (2006); "*A TFM document is complete if there is a relation for every output and the complete set of possible traces for which the value of each output is defined is included in the domain of the corresponding relation.*"

Ultimately the purpose of this methodology is to validate options for use within our case study construct of financial crime detection. While the restriction terms would inherently vary depending on the use case in question (AML¹/Periodic Fraud/Real-time Fraud) there should inherently be a consistency in approach to solve for verifiable specification patterns. Thus, we aim to produce a TFM component interface specification document that accurately describes a financial crime software use case from industry.

¹Anti-Money Laundering

2.5 Parnas Predicate Logic

As logic forms the basis to the validation work in this thesis it is important to understand the origins of the logic utilized in tabular expression trees. Parnas (1993a) introduces us to his predicate logic variant which is notable as it forms the basis for his other work in tabular expressions and document-driven design (Parnas, 2003a). The importance in this logic is placed in eliminating all forms of logically derived ambiguity hence making it a suitable precursor to the idea of tabular notation. With that in mind we can state for each assignment of values to variables that appear in a logical expression they should simply evaluate to either **true** or **false**. This ensures that no matter what inferred programmatic driven complexity exists prior to representation in this logic the outcome can clearly be tested, validated and understood with stark clarity.

An attribute that differs Parnas (1993a) from other forms of logic is the way in which it handles partial functions. A number of types of logic assume that all functions dealt with are deemed to be '*total*' functions. A *total* function means that the function is defined on a domain that includes all possible values of its arguments. A *partial* function on the other hand means that for a given function not all the possible values of the arguments on a given domain have been defined.

Mathematically we can see these as defined in Gallier (1986) where Gallier defines a *partial* function to be a triple $f = \langle A, G, B \rangle$ where A and B are arbitrary or even empty sets and G is a relation. Gallier (1986) continues noting that a partial function $f : A \rightarrow B$ is a *total* function if $domain(f) = A$.

Cheng and Jones (1990) introduces a survey of a few methods of dealing with partial functions using different logics. These methods include totality over a restricted set, viewing functions as relations, fixing notions of equality, conditional operators, to name but a few. Parnas (1993a) discusses another method of dealing with partial functions by altering the common perception of dealing with predicate expressions.

The use of Parnas (1993a) for a financial crimes definition should be a good application of the method. From a functional logic perspective the observed specifications thus far do not push too broadly into restriction terms dominated by partial functions rather, they attempt to simplify terms as much as possible for computational purposes. As such the specification types discussed by Bane et al. (2004) should be easily applicable to the context discussed and easily transition to proving stratagems for the mechanized aspect of the tool set. When this breaks down human interaction would be required to make a decision and finalize the resulting output.

2.5.0.1 Example Tabular Expression Tree

The logic written in the trace function method attempts to assist software engineers in constructing accurate specifications about their software. Janicki et al. (1996) builds on this work and incorporates this logic into an easily read form of document specification called the tabular expression or, simply put, a table. Tabular expressions allow software engineers to instantaneously note whether there are 'gaps' in their specifications as such 'gaps' are clearly evident when tabular expressions are used to specify software. Furthermore, because tabular expressions are constructed using predicate logic and other forms of mathematics, none of the usual ambiguities about some software are evident due to the elimination of natural language.

This mathematical model specifies how using a relational model applied to software that a tabular expression representing that software could be derived. Tabular expressions are useful because the same information can be represented using different tables thereby making certain aspects of the information more pronounced or clear. Due to this fact many different table types came into existence, each concentrating on representing data in a subtly different fashion.

Bane et al. (2004) introduces a new mathematical model for tabular expressions. Here the author introduces both an *evaluation* term and a *restriction* term. The evaluation term is clearly for use in evaluating tabular expressions whereas the restriction term gives us interesting semantic constraints being imposed on that specific tabular expression. The restriction is similar to what was proposed by Guttag (1977) which

explicitly notes when the value of an operation will loosely defined, but has been modified as per Bane et al. (2004) to map more with tabular expression notation.

The constraints on a tabular expression could be things like, type, cardinality or number of grids, for example. It is from the restriction term that we are given the means to construct a theorem as it contains all the 'facts' we need to know about that tabular expression.

Several types of tabular expression trees are utilized in Bane's mathematical model and we will examine some of the structural elements and table types in order to ensure that we have the full flexibility required to map a financial crime detection system to a tabular expression format.

Jin and Parnas (2010) defines an indexed set as being an indexed a triple (I, X, f) where I and X are finite sets and f is a function defined from I to X . Further, it's noted that the sets X and I can be of any type. With respect to the domain of the function f , I , is called the index-set; an element of I is called an index. Thus if S is an indexed-set, (I, X, f) , then $IndexSet(S)def = I$ or in an alternative representation if K is the indexed-set (I, X, f) and $i \in I$, either $K[i]$ or K_i can be used to represent $f(i)$. Either can be utilized at the preference of the person designing the specification.

Jin and Parnas (2010) introduced a normal function table as a popular example to explain the functionality of tabular expression trees. They are structured like Table 2.1 which describes a simple function, $f(x, y)$. In simple terms the function becomes x squared - y squared if x and y < 0. What follows is an examination of what occurs if you examine this more formally:

Jin and Parnas (2010) suggests that within the our example of a table construct there are several unique properties. They begin by first examining the concept of *constituents*. In the case of Table 2.1 there are several constituents to consider. They are:

- *integer : n.*

		T_1		
		$x < 0$	$x = 0$	$x > 0$
T_2	$y < 0$	$x^2 - y^2$	$x^2 + y^2$	$x^2 - y^2$
	$y = 0$	$x + y$	$x^2 - y^2$	$x + y$
	$y > 0$	$x^2 + y^2$	$x + y$	$x^2 + y^2$
		T_0		

Table 2.1: Normal Function Table

- $\{integer : L_i | 1 \leq i \leq n\}$.
- $grid : T[0], index - set = \{1, \dots, L_1\} \times \dots \times \{1, \dots, L_n\}$.
- $predicategrid : T[i], index - set = \{1, \dots, L_i\} | 1 \leq i \leq n$.

Jin and Parnas (2010) explains that the parameter n is the number of dimensions of T_0 that we see in Table 2.1. The parameter n should be noted as being always one less than the number of grids ($T_0 \dots T_n$) of the complete table. The L_i parameters give the dimensions of T_0 and the lengths of the one-dimensional grids, $T_1 \dots T_n$. In the example illustrated in Table 2.1, n is 2, the grids are T_0, T_1 and T_2 . T_0 is a 3×3 grid and both grids T_1 and T_2 have three elements.

Once the constituents of Table 2.1 are understood we next need to understand any conditions that they must satisfy, or to simply the *restrictions* associated with the example. In this case they would simply be:

- The number of grids in $T, n + 1$, must be at least 2.
- The lengths of grids $T[1], \dots, T[n]$ must be positive.
- Grids $T[1], \dots, T[n]$, must be proper¹ where *proper* is defined by Jin and Parnas (2010) through the example of a predicate grid, G , which is considered proper for a predicate C if, for every assignment of values to its variables that satisfies C , exactly one predicate in G is satisfied.

¹ $Proper(G, C) \rightarrow ((\forall j, k : j, k \in I, (j \neq k \rightarrow G[j] \wedge G[k] = false)) \wedge (G[i_1] \vee \dots \vee G[i_n] = true))$
 where $I = \{i_1, \dots, i_n\}$ is the index set of G .

Thus, we arrive at the final element of Bane et al. (2004) the concept of *evaluation*. For Table 2.1 a normal function table, T , represents a specific function that can be evaluated for an assignment, A , in two steps which are:

- Choose j_1, j_2, \dots, j_n such that $\forall i(0 < i \leq n)T[i][j_i]$ evaluates to true for A .
- Evaluate $T[0][j_1, j_2, \dots, j_n]$ for the assignment A to compute the value of T .

The explanation of this is that the value of this expression for an assignment A is the value of $T[0][select(eval(T[1], A)), \dots, select(eval(T[n], A))]$ according to Jin and Parnas (2010). Thus, for any assignment that is complete for T this holds true and as such, we need not mention A , and can simplify to the term.

$T[0][select(eval(T[1])), \dots, select(eval(T[n]))]$.

In this example expression, *eval* is used to indicate that the evaluation of the predicate grids must happen first and is followed directly by the selection of the expression in $T[0]$ to be evaluated rather than evaluate all expressions of $T[0]$ and then select one of the values. Jin and Parnas (2010) cautions that as there may exist some partial functions in the expressions, it may not always be possible to evaluate all expressions in $T[0]$ for every assignment. However in this case as predicates in the logic used are always total, we can evaluate any predicate grid for any assignment.

Prior to finalizing the translation of the above into a final TFM representation we need to define one more function that is highly utilized within the method; namely the *select* function. Using the same construct as the definition of the *proper* function we state that if PCG denotes the set of predicate Constant grids, *select* is a relation defined by:

$selectdef = \{(G, i) | G \in PCG \wedge G[i] = true\}$.

Jin and Parnas (2010) thus argues that when *select* is applied to proper predicate constant grids, it is a function and we can write “*select*(G)” to denote the index of the only element of G that is true.

Thus our TFM representation of Table 2.1 is;

Constituents:

- $integer : n$.
- $\{integer : L_i | 1 \leq i \leq n\}$.
- $predicategrid : T[i], index - set = \{1, \dots, L_i\} | 1 \leq i \leq n$.
- $grid : T[0], index - set = \{1, \dots, L_1\} \times \dots \times \{1, \dots, L_n\}$.

Auxiliary functions:

- *Proper* and *select* as per our definitions.

Restriction schema:

- $(n > 0) \wedge (\forall i, 1 \leq i \leq n \rightarrow L_i > 0) \wedge (\forall i, 1 \leq i \leq n \rightarrow Proper(T[i]))$.

Evaluation schema for an assignment that is complete for T:

- $T_0[select(eval(T_1)), \dots, select(eval(T_n))]$.

What we learn from this tabular expression example is that any method utilized needs to provide a broad range of flexibility for the definition of a mapped expression from a module specification. As developers attempt to use the most effective means of resolving problems there is no means to guarantee that they will solve the same problem in a consistent form. This means that we need multiple mechanisms to relate expression information to allow for deviations in how developers would construct an expression tree around their interpretation of the module interface specification.

2.6 Applying the TFM

The Trace Function Method of Parnas and Dragomiroiu (2006) provides a precise mechanism to describe software programs and the associated documentation of interfaces for information hiding components. As mentioned earlier, Parnas introduces the concepts of event, event descriptor, and trace and provides definitions. Further, basic functions on event descriptors and traces are introduced in order to simplify its utilization for developers and or documenters.

Quinn (2007) discusses the classical application of explaining a formal method by using the TFM to describe a stack program example. However, other examples for the explanation of the method exist and are equally illustrative in showing the flexibility of the method. The importance of any example is to provide a comprehensive full specification which as defined in Parnas and Dragomiroiu (2006) implies that a full specification is any specification which states *all* properties. The emphasis highlights that many specification methods focus only on required properties associated with the product description.

Earlier in this document we discussed some other approaches to describing software which try and meet the needs to describe the interface specification without compromising the internal operations of the program. Parnas and Dragomiroiu (2006) and Quinn (2007) both create category groups associated with these other approaches which in their terms loosely collect to being pragmatic, algebraic, axiomatic and a mix of two or all approaches. The conclusions drawn are that each approach has a discrete set of strengths and weaknesses associated therein. Extrapolating from that work, the major weaknesses that leap out are the lack of intuitiveness of the specifications applied in each method, the inelegance of the description and the limitations in applicability.

The TFM is interesting as it brings new elements that can be utilized in the goal to provide accurate, elegant and complete descriptions of software specifications. Parnas and Dragomiroiu (2006) discusses these at length but in brief they are:

- The TFM is neither equation-based or axiomatic.
- It is flexible in the number of multidimensional tabular expressions it supports.
- It utilizes the logic Parnas introduces in Parnas (1993a) which provides an elegant means of dealing with partial functions as opposed to other higher order logics.
- The TFM provides support for modules that use globally referenced variables in their communication.
- It provides clarity in the conveyance of important information pertinent to the module.

- It provides an abstraction layer from the internal operations of the module.
- It provides clear delineation between important information essential to a module's operation versus non-essential elements.
- The TFM builds on existing mathematical concepts.
- The TFM is very specific in how information is organized.

One of the major elements of Parnas and Dragomiroiu (2006) is how it defines a module as a finite state machine and the temporal evolution of its states linearly through a series of events which potentially read or update variables and alter the internal state of the module. Further descriptors are provided for these events such as *PGM* The Name of the Program invoked at a specific event. Descriptors examine variable states through changes in their values. Specific treatment of input and output variables utilize abbreviated descriptors.

Traces are of vital importance to the method and as explored formally in Parnas and Dragomiroiu (2006) we understand that it is a grouping of event descriptors that captures a certain sequence of events allowing for further nuance through the use of sub-traces.

The TFM thus requires its interface documentation to have complete specifications of a program's inputs, outputs and relational information tying the operations on inputs to output values. Completeness in the TFM is defined by Parnas as being when there is a relation for every output defined and there exists a complete set of traces for which the value of each output is defined and included in the domain of the corresponding relation.

As such, our goal for our industrial application for a financial crime system is to provide a TFM description of all modules such that all possible traces within that program are specified. Through this specification we hope we will be able to verify the completeness of the program and the accuracy of its documentation.

To simplify the description of industrial programs Parnas and Dragomiroiu (2006) provides a series of primitive functions on events and basic functions, predicates and function generators for use on traces.

It is important to note though that as the examples provided by Parnas and Dragomiroiu (2006) are quite simple one should consider more complex examples like Quinn (2007) in order to gain a broader perspective on the application of the method within a more complex finite state machine example. While the trace function method does not become a silver bullet to describe with ease all aspects of formal specification of module design it does provide a more holistic and pragmatic approach to doing so. Our conclusion here is that while a myriad of other methods exist that attempt to solve the problem only the TFM has the potential depth of mathematical validation necessary to ensure a complete and accurate description providing the necessary coverage for complex critical systems. We will attempt to validate this assertion later in this document.

This conclusion is shared and validated through examination of works like Semegn (2011) where a more comprehensive analysis of the approach is conducted specifically around ensuring consistency in specifications.

2.6.1 Tooling Supporting Tabular Formal Methods

Some of the previously discussed approaches have attempted to create tooling to reduce the burden on software developers to enable themselves on the approach. A lot of emphasis has been put on developing tools that have been adapted to map to the approaches that originated in Parnas (1972*b*). The evolution of this tooling has been to move from standalone tools like Heitmeyer et al. (2005) and move to more integrated applications or widgets like Eles and Lawford (2011). Given our previous discussion on the tabular notation itself it is clear that any tooling would provide significant lift for software developers using formal methods. This lift would come in the form of mechanisms to input, output, verify and store the representations easily and potentially should integrate with development environments.

Of the above mentioned research into this area associated with Parnas-based methods, each attempted to create some form of tool suite to interact with tabular expressions. However, not all of them were successful in creating a complete offering of the components necessary. Only the McMaster Jing (2000), NRL Heitmeyer et al. (2005) and the OPGI Lawford et al. (2004) research teams were able to create tooling that included mechanisms to design, check and output tabular expressions. Other research did include some capabilities that focused on evaluation of expressions and some manifestation of a kernel but did not attempt to make a more complete offering for end-to-end program descriptions.

The McMaster TTS and more recently the table toolbox for Matlab/Simulink Eles and Lawford (2011) would be seen as immediate predecessors for more recent versions of the table tool system. It was designed and implemented over a period of years by both staff and graduate students at McMaster University in Ontario, Canada. The first iteration was the TTS and it was composed of three layers of operations;

1. The Table Holder & Information Modules, the basis for all logic data structures and persistence layers for tabular expressions.
2. The Library Layer provided abstraction information to the tool layer.
3. The tools themselves, which ultimately contained a table checking tool amongst other user tools for the manipulation of tabular expressions.

In the next generation of the McMaster research work they took a very different research path, opting instead to focus on integration with existing toolkits to provide the same level of functional capabilities. As such the division of functionality aligned as follows:

1. Matlab, providing the library of cohesive end user capabilities to graphically edit, generate and manipulate tabular expressions.
2. Simulink, to gel all the component layers together to provide a seamless end user experience for designing and verifying tabular expressions.
3. PVS and CVC3, two automated theorem proving systems to provide a mechanism to verify tabular expressions themselves.

The use of the theorem proving system PVS, Prototype Verification System Owre et al. (1992), for both approaches makes sense given that PVS has existing support for tabular expressions. This allows the research teams to focus instead on generating PVS code directly to support any given expression tree and the PVS system generates the subsequent obligations surrounding disjointness and completeness conditions.

CVC3 is an automatic theorem prover for Satisfiability Modulo Theories (SMT) problems that can be used to prove the validity (or, dually, the satisfiability) of first-order formulas in a large number of built-in logical theories and their combination (Barrett and Tinelli, 2007). It is a very different entity to PVS in terms of approach and does not inherently have the support necessary for tabular expressions. Thus the McMaster team needed to extend their toolkit to generate the obligations for disjointness and completeness conditions for CVC3 in query form. The generated queries can then be provided to the proof stack of the CVC3 tool and it can thereby attempt to verify the queries. It outputs a counter model post evaluation, should it consider a provided query to be invalid in nature.

When considering the basis for both of the above approaches (complete development versus integrated development) it is important to consider the variations of all tabular expression tool suites surrounding table types management. We agree with Bane (2008) on their commentary regarding table type extension to support model flexibility, which is something that the McMaster team succeeded in doing in both efforts at designing tabular expression toolkits. This gives both these efforts a distinctive edge over the NRL efforts which really had hard-coded table types built into their toolkit. McMaster seemingly built on the work of Janicki et al. (1996) in successfully showing a mechanism to provide broader flexibility for type manipulation.

As the McMaster TTS utilizes a formal model with a library of each table type developed as independent segments of code, it provides a formal layer of abstraction that provides a common definition of general tables with specific code basis for variations. This abstraction layer helped in providing Jing (2000) a solid basis for deriving PVS code based on these table definitions that could be in turn checked for the appropriate level of completeness and disjointness.

Due to the nature of the NRLs core purpose as a military research facility, the manifestation of NRL SCR* tools are being used on military problems such as guidance systems, weapons control systems and power governance systems. The SCR* tools are discussed in the supporting literature for the research project Heitmeyer (2001) and Heitmeyer et al. (2005) and specifically reference work around designing, checking and outputting tabular expressions.

With the aim of improved accuracy and rigour the SCR* tool suite made several attempts at designing and implementing tooling for tabular expressions. On examination of the checking mechanisms specifically, they varied in a number of ways depending on the depth of verification, programmatic complexity of the associated tabular expression tree as well as the integrated back end mechanism to automate the validation effort. Like the McMaster effort PVS was also utilized in this research, mostly because of its support for tabular expressions and also because of its accuracy in the associated proof obligations. Model checkers were also examined but had limitations in scope, as model checkers at the time were highly limited in breadth of function in comparison to theorem proving systems like PVS (SPIN Holzmann (1997), Salsa Bharadwaj and Sims (2000), PVS Owre et al. (1992)).

These capabilities are able to eliminate a lot of the tedium associated with checking tabular expressions but were not entirely capable of checking all conditions and expression trees passed to them. As such the NRL project also made use of human review to ensure the accuracy of the results thereby undermining the automation elements associated with the system. To conclude, the overriding factor in determining the success of the NRL SCR* toolkit was based on the limitations of the associated verification technologies and not the tools themselves. Usability and interaction design were elements of concern but ultimately these were out of scope for the project. Nevertheless, they attempted to utilize methods to ease interactions with these systems to promote a healthier utilization of the checking methods Archer (2000).

Like the McMaster and NRL efforts, the OPGI (Ontario Power Generation Inc. Lawford et al. (2004)) developed a set of tools that was instantiated in design to assist in

formally documenting the design of nuclear reactor shut-down software systems Lawford et al. (2004). Like the other research efforts here also PVS was heavily utilized as the back end validation systems. This highlights a key common factor in research efforts we've discussed which is a dependency on the PVS automated theorem proving systems. PVS is not without its problems and limitations so it is very interesting to note the continued use of it as a basis for these efforts. We've determined that a number of factors led to this, mainly the lack of mature competitor capabilities. Potentially model checking or computer algebra systems could have been utilized but at the time they did not offer the same breadth of function. Further, they also did not offer the native interfaces for tabular expressions via the TAME project that PVS offered Archer (2000).

In Jing (2000) we garner further insight in the limitations of the McMaster efforts where evaluations on the PVS system led to a number of conclusions on its effectiveness. Namely, Jing focuses on a number of function related issues associated with the PVS proof function library. Jing conjectures that the *GRIND* function in particular is highly limited due to its reliance on a specific tuple structure. The grind function is defined as being the highest-level automated proving strategy in PVS. It does skolemization, heuristic instantiation, propositional simplification, if-lifting, rewriting, and applies decision procedures for linear arithmetic and equality Shankar et al. (1993). This issue with *GRIND* is still the case and as such predicate tuples with different types of predicates may be unsolvable due to the expression structure. As a tabular expression checking tool system is envisioned as being flexible and supportive of multiple predicate statements, this inherent limitation could lead to forced obsolescence of the toolkit.

Jing evaluates the usefulness of *GRIND* in general terms throughout the research into theorem proving, as generally it is faster and capable of solving *most* problems passed to PVS for checking. Regardless of its issues, even Jing agrees that PVS offers alternatives to solve complex predicate statements like vector tables and avoid skolemization, iterative expansion and heuristic methods as a solving strategy which generally cripple alternatives as each presents a logical problem that *breaks* the proving ability

for complex specifications. These issues are common amongst many of the mechanized mathematical tools available but work-arounds exist in some situations.

Other limitations of the McMaster variants of checking tools vary depending on the generation observed. In the MinJing variant there was a statute of limitations associated with supported tabular expression tree types, meaning that a number of table types were specifically not supported by design. The following table types were supported by the system:

1. Normal Tables
2. Inverted Tables
3. Vector Tables

When examining the later generation of the tool kit this appears to have remained consistent with the research team making a scoping decision to not support special cases like decision tables.

Further limitations included determining the uncovered area of a tabular expression should the domain of the table be incomplete which leads to issues in the underlying mathematical structure surrounding partial functions. External to these factors, all other limiting factors were purely technical in nature focused on infrastructure and communications abilities between the other table tool suite components. This in part was solved for in the next generation of the suite as it successfully removes operational complexity by integrating existing systems to simplify operation interfaces. This does however introduce the potential for misrepresentation of specific expression assets when parsed between different component libraries.

Other examples of the use of tabular expressions include innovative work introduced in Singh et al. (2015) which looks at a case study of Insulin Infusion Pumps. This is another example of a critical system application of the method to ensure accurate descriptions of the software code and insure the accuracy of the document driven design of the program used for the industrial application. The main contribution of the paper

however is a refinement strategy to aid in the automation of formalization of requirements from their associated tabular expressions which the authors define as being a *correct-by-construction* approach.

To impart the effectiveness of the approach they use the **Event-B** modelling language as a basis for the industrial application. This is a departure from previous efforts which utilized automated theorem provers for this function. These representations can be thus used to check for correctness of the functional behaviour of the program with the associated properties provided. The authors conclude that they can guarantee that the safety properties are preserved throughout the life-cycle of the associated program thereby ensuring the necessary critical elements for safe operation of the insulin Infusion Pump.

While it is clear that some efforts have been made to develop tooling the majority of these have been driven by academic initiatives. There has been some success to ensure that there is a suite of available tools for verification, inputting and outputting tabular representations but none of these has been part of or extension to an integrated development environment used in industry. One could argue that Eles and Lawford (2011) attempts to bridge that gap but the widget created here only addresses a very targeted audience. Later in this thesis we will examine the impact of this lack of tooling and discuss the implications for the method in a financial crime context.

2.7 Summary

This chapter examined the related work in the field of formal methods. It examined several different types of methods employed in critical system design documentation Dehbonei and Mejia (1994), Abrial (1996) and Bartussek and Parnas (1978). Further it delved into tools developed to support formal method applications in Jing (2000), Lawford et al. (2004), Heitmeyer (2001) and Eles and Lawford (2011). It looked for commonalities in approach and what industry applications the tool was utilized for.

Later the chapter introduced several preliminary elements surrounding the Trace Function Method (TFM). It examined elements of the historical approach around traces and discussed elements utilized by the method in order to create comprehensive descriptions of modules. It examined work in this area like Bane et al. (2004), Parnas and Dragomiroiu (2006) and Quinn (2007) and discussed some of the definitions and examples provided.

The chapter showed that while simple examples should have simple descriptions increased complexity would generate the same degree of complexity in their descriptions and that no method could alleviate that concern. Effectively while the TFM approach has areas of coarseness it represents the best means of generating accurate descriptions.

"For the powerful, crimes are those that others commit."

Noam Chomsky

3

Use Case Legalities

3.1 Introduction

The evolution of policy and laws around customer due diligence functions originated due to a widespread utilization of criminal agents of the banking system. The integrity of the financial system globally was challenged by regulators internationally and thus new regulations were drafted based on policy makers advice. One major policy maker FATF (Financial Action Task Force) and others lobbied for drastic change to how banks operate in order to highlight accounts owned by criminal organizations, dictators and terrorists. Prior to regulation of this nature institutions were able to offer anonymity products which could be used for nefarious purposes by their clients or by ultimate beneficial owners of the accounts (Tang and Ai, 2010).

In this chapter we will examine the high level provisions recommended by FATF for the purposes of accurate customer due diligence record keeping. Further we will look at the realization of the policies as applied to the fourth EU directive on anti money laundering. Money laundering is a tool utilized in order to obfuscate wealth and the movement of funds internationally. Whilst commonly utilized by organized crime rings it is more often utilized today by individuals associated with corrupt practices or their agents (Siclari, 2016).

Even with increased legislation and technological process to detect money laundering the practice is still quite rampant. Criminal intent still attempts, and succeeds, to circumvent controls within the banks themselves often using the guise of legitimate

business to achieve this (Riccardi et al., 2016). Commercial trade is the perfect instrument to hide money laundering as the volume of cash transactions involved is huge and import/export documentation can be forged especially when combined with corrupt practices like bribery.

3.2 FATF

The Financial Action Task Force (FATF) is an inter-governmental body that was established to promote mechanisms to counter both terrorist financing and money laundering. The mechanisms commonly utilized tend to be driven by legal, regulatory and operational means but FATF remains a policy making body that drives for the reforms necessary at a government level (FATF, 2012). As a policy-driven organization FATF is unique in their employed methodology in the fight against terrorism globally (Gardner, 2007).

The majority of the policies FATF promotes has been aggregated into a series of specific recommendations that tackle a portfolio of anti-money laundering and counter terrorism financing operational needs. Of particular interest to this research is their recommendations for customer due diligence (CDD). In their recommendations FATF provides an exhaustive set of guidelines specific to CDD and enhanced due diligence (EDD). These recommendations are pushed not just towards the adoption by the EU but also the G8 countries (Scherrer, 2006).

The foundation of this recommendation is scrutiny applied to business relationships, transactions that exceed a fifteen thousand dollar threshold or suspicious wire activities. Historically, the integrity of the financial system could be compromised when an organization allowed the proliferation of accounts that did not have adequate data collection methods associated with the account holder. Examples of this included cases where anonymous accounts are held (Austria, Switzerland) or the utilization of fictitious names to mask true identity.

Such measures became necessary due to activities conducted by individuals and governments around wartime, dictatorships or state sponsored terrorism. As such the banking sector had doubts about the validity of account ownership and how the funds were being utilized (Badali, 2010). With these measures enacted regulators now have a legal instrument to punish institutions found wanting in terms of their operation compliance risk management processes.

When the recommendations are enacted into national regulatory compliance laws, banks must find means to operationalise the legalities, this generally means utilizing various technologies to ensure compliance. While many countries have agreed to participate there is an active list of countries that choose not to enact anti money laundering and counter terrorism financing measures into law. As such FATF maintains an active list of high risk and non-cooperative jurisdictions. This list has implications for banks that are monitoring client records and transactional activity.

Thus it is incumbent upon a bank in a participating country that it must do as follows (FATF, 2012):

- Verify customer identity utilizing reliable forms of documentation.
- Seek out beneficial owners and take reasonable means to verify their identity.
- Understand the nature of the business relationship between the beneficial owner and the client.
- Apply ongoing due diligence on the business relationship and scrutinize transactions to ensure that they are within the risk profile applied to the client relationship and the source of the funds utilized.

The approach further advocates the use of a risk-based approach where the bank is responsible for creating a means to stratify their client populate in terms of risk buckets; generally high, medium and low. However, as explored by Fratangelo (2016) there a number of different means to create various custom style risk strata depending on the products/channels a given institution operates within.

In instances where an institution is unable to acquire the necessary validation elements like passport, national identification numbers etc. it is incumbent upon them to not initiate a business relationship or open an account for said client and potentially file a regulatory report regarding the customer for national security agencies or regulators to examine and investigate. The requirements here dictate that all clients, new or existing, should be subject to this form of continuous risk assessment to ensure against malicious activity.

Given this burden upon them it is not surprising that financial institutions have turned to technology to help them with this risk assessment process. The record keeping requirements have transitioned from traditional printed duplicates into electronic forms stored in data warehouses. Institutions are required to keep a five year rolling window of all interactions with a client with the associated audit information up until five years after the termination of the client relationship.

An ancillary requirement of these policy recommendations is the capture further measures in the case of foreign politically exposed persons for customer due diligence purposes. These are:

- All institutions must have operational means to uncover using a risk based approach if the client or beneficial owner of an account is a politically exposed person (PEP).
- Approval from senior management to establishing or continuing a business relationship with a PEP. In some geographies individuals may have personal liability when they obfuscate the details of client relationships under AML law.
- Establish the source of wealth for the client.
- Conduct enhanced ongoing monitoring (Enhanced Due Diligence) over the course of the business relationship.

The diligence requirements applied by banks on PEPs is also applicable to their relatives and close associates. This is as a direct link to the use of organized crime of family members to mask the movement of funds for money laundering purposes as explored in Serio (2008) where mafia elements utilize relatives and close associates to launder money across several geographies.

3.3 The Fourth EU Directive on Anti-Money Laundering

The European Union over the past decades has enacted several directives with regard to anti-money laundering and counter terrorist financing. Most recently on 26 June 2015, the 4th Anti-Money Laundering Directive (EU) No. 2015/849 (4th AMLD) entered into force. It is incumbent upon all EU Member States to implement the 4th AMLD by 26 June 2017 into national laws and regulations (EU, 2012).

The 4th AMLD builds upon the existing 3rd Anti-Money Laundering Directive (Directive 2005/60/EU (3AMLD)) and the directive that drove its implementation (Commission Directive 2006/70/EC). The driving force behind it is the adoption of new recommendations released by the Financial Action Task Force (FATF, 2012).

With the 4th AMLD more emphasis is accorded to the activities of a risk-based approach to enforcing adequate protections for the the detection of suspicious activity or irregular account behaviours. The EU organizations responsible for measuring compliance with the directive will conduct an omnibus analysis of money laundering and terrorism financing risks as applicable to all its members. The findings of this analysis and subsequent recommendations were sent to the respective countries under EU membership. Armed with these recommendations and the analysis of the risks the membership is able to apply tighter controls to its national financial regulations. Effectively, this represented a starting point as each country may have unique risks that cater towards some more economic driven factors (resources, borders, trade) and the

expectation is that each member state ensure that they have adequate risk coverage as per the FATF recommendations to meet the unique elements also.

Due to increased concern regarding beneficial owners of accounts, the 4th AMLD directs that countries in the EU create a registrar that has a holistic index of beneficial owners of corporate entities. To adhere to this the financial institutions under the mandate of regulation look at statements of corporations along with information provided by corporations themselves regarding its director and business area through taking customer due diligence measures. A centralized register is intended to be a more reliable and comprehensive data repository that can be used to identify potential beneficial owners. Further, with such an active registrar in place at a national and financial investigation unit (FIU) level transparency levels applied to beneficial ownership should improve meeting the explicit demands of the FATF.

Through its implementation, (EU, 2012) provides justification that national regulatory authorities and Financial Intelligence Units must have access to the central register under the reciprocating member state national anti-money laundering legislation for exercising their customer due diligence law. To simplify, this means that each country can maintain a central database that banks can query regarding citizens to understand if the information they provided regarding source of wealth or identification details is accurate. It is feasible that other organizations or even individuals may be allowed access to the registrar in cases where they can prove the legitimacy of the request but meet exceptions in the form of information regarding trust structures. Each individual EU Member State should be entitled to restrict access to certain fields of sensitive information regarding beneficial owners completely or partially in exceptional cases where they feel there is just cause regarding the request.

As the key element is information and record keeping, ensuring the quality, standardization, normalization and storage of these elements is key (Angelopoulou et al., 2012). As infrastructures to enact the directive are implemented a core aspect of technologies utilized will be the ability to provide accurate logs on all customer touch points over the client life cycle. This is to provide regulators with effective means of governance on the bank's execution protocols for the directive.

As part of the directive there is no longer any form of differentiation in EU (2012) between politically exposed persons (PEPs) residency, be that their naturalized country of residency versus other countries. There are other obligations that apply to PEPs classified as beneficial owner of different financial products/accounts. EU (2012) expands the category of PEPs to include members of the governing bodies of political parties. At an operational level the additional PEP designations will result in the necessity to update existing PEP lists, thereby increasing the operational requirements for banks around Know Your Customer (KYC) compliance.

Under the existing regulations the financial services sector is already obliged to meet with regulatory compliance system requirements and the 4th EU AML directive (EU, 2012) simply builds on this pre-existing set of measures with more strictures based on customer due diligence and beneficial ownership legislation. As this is being enacted as an EU wide initiative it will enforce action for all institutions and their subsidiaries, both within and outside the EU, to comply with the enhanced measures. Failure to do so will result in the potential application of severe penalties for institutions deigned to be operating outside of regulatory compliance.

3.4 Conclusion

With FATF (2012) we have a credible reference that can act as a basis for modelling customer information and financial transactions for the purposes of customer due diligence. The data attributes discussed in the 4th EU AML directiveEU (2012) builds upon this foundation but allow for flexible adoption for EU member states. Globally, there is a variance in adoption of the recommendations with some countries opting to partially comply or remain completely outside of compliance.

Technology vendors can use these regulations to enact an operational facility to detect money laundering and terrorist financing whilst giving some further utility around data representation to the banks they service. As compliance is the most critical element of this problem transparency around scoring elements is vitally important. Thus, underscoring the need for specifications of function and score that are clear, concise and exact.

"Computers are useless. They can only give you answers."

Pablo Picasso

4

Financial Crime Detection Technologies

4.1 Overview

This chapter introduces the idea that financial crime systems should be treated like critical systems. It further surveys several key technologies in use to provide solutions to the different types of financial crime pervasive within the banking sector.

Examples are provided to explain the complexities associated with describing financial crime detection programs and an analysis will be conducted on the appropriateness of using formal methods to help alleviate concerns around adequate documentation for regulators.

4.2 Financial Crime Detection as a Critical System

Financial systems are dominated by mechanisms that protect and regulate its institutions from monetary loss the world over (Phua et al., 2004). Increasingly, these institutions are the victims of targeted attacks by individuals and groups that seek to exploit weak detection systems or the vulnerability of employee/colleague trust for profit (Vanasco, 1998). Subversion of financial systems by criminal gangs to launder the proceeds of crime or by terrorists to fund acts of terror are common practices that consistently need urgent attention and action to counter. This area has a considerable

impact on all fiscal endeavour and can have as a result a critical impact on our lives. Thus I assert that such systems by definition are representative of critical systems and should face the same degree of rigour and specification as examples from military and nuclear applications.

Criminal activity is ever evolving globally with criminal organizations utilizing new techniques and technologies to advance their goals and circumvent security systems. As a result, these systems are also forced into rapid evolution cycles to counter such schemes (Phua et al., 2004). We propose an alternative approach utilizing better mechanisms to specify and control the design and outputs of these systems.

Commercially there are many systems which utilize varying techniques to counter criminal activity using business rules, modelling languages and specially-designed strongly typed structural languages. Ultimately we are interested in helping to stop criminals and terrorists successfully finding holes that can capitalize on. For this research we are reducing our scope to solutions focused on data pattern detection systems using data science techniques discussed by Zhang et al. (2011), Jyotindra and Patel (2011) and Behdad et al. (2010) as these represent the largest cross-section of modern techniques to detect these form of crimes. In order to craft a methodology around studying financial crime detection systems using tabular-expression-related technology we must first examine the feasibility of translating the specifications used in these domains to a formal methods specification language like Parnas (2003a).

Should these systems be a successful application for formal method representations then regulators and the financial services sector will have a secondary mechanism of ensuring compliance with national regulatory legislation and still be able to scale appropriately as new payment channels become available. We define success in this case as the reference documents being able to act as independent entities that are able to prove or disprove compliance with a specific aspect of AML regulatory compliance.

4.3 Financial Security System Specifications

It is clear when examining the work of the OPGI team (Lawford et al., 2004) that design through documentation as recommended by Parnas (2003a), provides a mechanism to assuage concerns around software flaws, gaps and inaccuracies in design. When we consider financial crime system design however, these too have evolved due to regulatory and fiscal drivers resulting in software that attempts to meet those laws but invariably provides an abstraction layer to them due to the interpretation of laws that they are attempting to create solutions for. As such it should come as no surprise that within the sector of finance this issue persists as an ongoing problem which costs financial institutions of all sizes and business areas considerable sums of money annually in order to validate the accuracy and compliance of these systems.

The issue is that this type of behaviour in today's global economy cannot be summarized as the lone act of sole individuals; rather it has expanded to include organized groups targeting weaknesses in our institutions and acting on these weaknesses internationally. As such the scale of this activity cannot be ignored and countermeasures have evolved through both the stimulus of financial loss from penalties for non-compliant banks along with a growing regulatory commitment from governments worldwide. Thus we argue that the specification problem has to account for a number of criteria to aid and not hinder efforts of detection, namely:

1. **Opportunistic Detection Specifications:** Driven more by logic and detection of spurious activity outside an observed set of normalized behaviours (Denning, 1987).
2. **Organized Detection Specifications:** Driven by models that are able to resolve and link data attributes and entities for the purposes of risk scoring (Lades et al., 1993).
3. **Scalability:** Volumes of information are high in this use case and as a result a specification checker would need to be able to cope with excessive data volumes (Aghion and Howitt, 1990).

4. Type Support: Due to the different data storage mechanisms and variance of function in these systems any checking tool engine would need the flexibility to support multiple types and scale to provide mechanisms to check new ones. Bane (2008) discusses this at length and highlights flexibility as a major concern in any tabular tool kit.

There are a wide variety of technologies in play in any given institution at one time (Phua et al., 2004). Also it is not unusual for a given institution to place multiple layers of security and analysis around core processes in order to facilitate better governance and monitoring. As such it should come as no surprise that financial institutions represent a key driving force in the software engineering industry.

Logically one can extrapolate the basis of securing an institution from fraud and other forms of criminality in the same way one could think about the problem of intrusion protection for these self-same institutions. The analogy being that for a given perimeter, in this case the banks assets, we can envisage that bad actors are at play to both penetrate the layers of defences, whilst simultaneously there are good actors trying to protect against that action: hackers versus defenders.

The technologies behind these systems utilize the following techniques: data analysis, profiling, machine learning and neural networks. As such they tend to be adaptive entities which are easy customizable in the face of an ever-changing enemy. The systems themselves adapt internal security models to protect and secure investigative data from prying eyes and often integrate with existing bank systems to enact as much protection as possible. The inherent requirement here is that any form of formal representation must be easily maintainable to account for these extensions and adjustments.

The institutions demand numerous things from these systems. Primarily though, the bank is interested in being a profitable entity that does not bleed cash unnecessarily. One of the more obvious means of losing money is through criminality, however these institutions also can lose considerable sums to non-compliance with national financial regulations.

As big business entities themselves, the banks and insurers are also interested in value for money and as such are continuously looking for products with more features that are cheaper, easy to implement and learn.

So we can state that the institutions specifically hunt for an internal security system that:

1. Is affordable: A system must represent value for money in that it must cost the bank less than current losses or be a fraction of impending fines (English and Hammond, 2014).
2. Is adaptable: Criminals are always finding new means of taking advantage of financial systems. Any detection security system needs to be able to quickly respond to new types of threats once the threat becomes noticed (Gao and Xu, 2009).
3. User friendly: The traditional fraud investigator is not very tech-savvy and, as a result, any such solution needs to display data so that its quickly absorbed and evaluated by even the least technical of users (Senator et al., 1995).
4. Is Reliable: High availability is a consistent demand of modern enterprise and banks are no exception to this. Criminals can strike at any point and the institutions need to be able to react. Further, the demands on these systems are growing as the complexity of the crimes increases. As such, the stability of a given environment is of critical concern (Graham and Lussier, 2012).
5. Is Scalable: The systems have to scale both horizontally and vertically in function to deal with functional advances on the vertical to tackle new crime types and horizontally to deal with expansion of their clients customer base or reference data (Rajan, 2010).

The systems themselves are built to examine the *fiscal* and *static* data of a given institution. We can define *fiscal* to include any form of monetary transaction, be that a deposit, withdrawal, credit card transaction or a claim payout. We can define *static* data to be any form of customer reference data which describes the customer, their account, associations data and supporting data e.g. passport information, driver license, etc.

The majority of systems in today's marketplace rely on a combination of both types in order to derive decision data. We define *decision* data to be any data which shows and supports an argument that a detectable criminal action has occurred for a given entity or group of entities (Phua et al., 2005).

How systems get that decision data depends, as each utilizes different techniques to arrive at that endpoint. Later in this chapter we will examine the mathematics involved in reaching that point, however we can describe the techniques they use. One type of system is based on the principal of learning behaviours for a given entity. As such over time the system learns what constitutes normal behaviour. Using a series of rules entered into the system, through a code level application programmable interface or via a user interface, it highlights information that does not follow that learned pattern (Bolton and Hand, 2001).

Another type of system is one where a static snapshot of the data is taken and fed into an analysis engine which uses a library of existing abnormal/criminal behaviour patterns and looks for matches (Ghosh and Reilly, 1994).

Other systems utilize the concept of social networks to trace pockets of activity highlighting potential accomplices and dubious actions and entities. A dubious entity can be a number of different things but normally is a phone number, address or something along those lines that can be used to link criminal activities (Goldberg and Wong, 1998).

As these systems continue to evolve, it is not unusual to have hybrid versions that incorporate some or all of the above concepts.

The differences between this domain and intrusion testing tend toward the separation of concerns around data. Holistically we can diversify by stating anti-fraud and anti-money laundering systems examine inferences acquired through the understanding of data and the relationships between all data elements held therein. On the other hand, intrusion testing centres on evaluation the susceptibility of a given data store is to unsanctioned access (Denning, 1987).

Ergo, it is the purpose of any enterprise financial crime solution to protect and understand the data held in the enterprises infrastructure. For the most part in the commercial world these solutions are divided in purpose due to the functional realities. Security access to data clearly permeates throughout the banks infrastructure, however for the most part intrusion-based system concentrate around external interfaces where the possibility of unsanctioned access is most probably. This is further enhanced through the utilization of further governance mechanisms to ensure access control lists for critical systems and data stores.

Historically, fraud detection systems have relied primarily on business rules. This has been good for identifying re-occurrences of lessons that have already been learned. There are three main issues with utilizing only this methodology. First, business rules create a lot of noise as legitimate customers constantly do things that look suspicious but are likely not criminal. Examples would include depositing a cheque with a greater than average value, submission of an insurance claim, changing their address, adding a spurious payee to their on line banking. False positives take time to triage and result in operational inefficiency (Phua et al., 2005).

Second, business rules eventually become common knowledge to the fraudsters due to either trial and error in fraud schemes or worse, infiltration of the organization causing business rules become known. This results in a risk to your organization, which results in constant tweaking of dollar thresholds in the associated scenarios, which result in more operational inefficiency. Finally, business rules are not forward looking. They are not there to catch tomorrow's fraud types (Bolton and Hand, 2002).

What an approach utilizing advanced analytics offers is a methodology that helps counter the problems that a business-rule-only approach fails to address. Using the concept of risk factors we can begin to move into a world where we are dollar amount agnostic. Organizations should not be forced to only try to catch the fraud over a certain dollar amount. All fraud losses are going to the bottom line and in today's economic climate every dollar counts (English and Hammond, 2014). Secondly, an advanced analytics approach looks for true fraud anomalies versus the narrow scope typically associated with business rules for specific fraud schemes. Finally, Social

Network Analysis (SNA) brings to the table a new tool for detecting and visualizing fraud. Too often investigators are so focused on individual cases that they do not see the forest for the trees. SNA is about discovering previously hidden relationships that are meaningful to an organization. In the list below we itemize the approaches and describe them:

1. **Predictive Modelling:** Perform knowledge discovery, data mining, predictive assessment based on previous disposition of alerts and cases. Examples of techniques used would include neural networks, decision trees, generalized linear models, econometric models and gradient boosting (Phua et al., 2005).
2. **Text Mining:** Unlock the power of unstructured data within reports, staff notes, and websites (Bolton and Hand, 2001).
3. **Anomaly Detection:** Identify individual and aggregate abnormal patterns that exist within your data. (Mean, standard deviation, percentiles, univariate & multivariate regression, clustering, sequence analysis, peer group analysis)
4. **Automated Business Rules Using Machine Learning):** Automate rules based on known patterns/factors leading to suspicious behaviour using machine learning and filter those transactions. An example would be two transactions in different time zones from the same account owner within an unreasonable short period of time (Bolton and Hand, 2002).
5. **Database Searches:** Accessing internal and external data sources like data vendors such as Lexis Nexis, CIS, Web search, Dunn/Bradstreet and OFAC.
6. **Social Network Analysis:** Establish connections between people and businesses through associative linkage analysis. Examples include social network, linkage analysis, community detection and advanced analytics (Goldberg and Senator, 1995).

The long-term goal of a bank to solve financial crime concerns is to establish a framework for enterprise-wide deployment of resources, including both material and human assets. This framework should make it possible to:

1. Gather and cross-match relevant data from all product lines, organizational units and geographic regions of your organization (English and Hammond, 2014).
2. Analyse this data to *connect the dots* and spot large-scale fraud attacks early in their life cycle (Denning, 1987).
3. Plan and execute focused countermeasures to combat large-scale attacks.

There are two key business drivers that are causing organizations to give serious attention to an enterprise-wide strategy. These are:

- Increased effectiveness. The ability to look at the issues holistically across the enterprise and identify large-scale threats early in their development, and mount effective countermeasures while there is still time for them to have maximum impact.
- Increased efficiency. The ability to leverage investments in data, tools and personnel in an economic environment where every organization and function is being asked to do more with less.

4.3.1 Detection Techniques

Bolton and Hand (2002) applied critical thinking to the industrial field of fraud and financial crime, the result of which was a definitive work that outlines the systematic approaches undertaken to detect anomalous activity specific to fraud within transactional data feeds. The methods discussed centred around two distinctive approaches: supervised and unsupervised.

Supervised methods generally confine themselves to the forensic element of detection. This means that, in order to be effective, a seeded data set of both positive and negative events needs to be provided to the engine in order to determine with any degree of certainty that fraudulent activity is potentially occurring in the rest of the data space. The mainstay idea being primarily that, with a set of *known bads*, it is feasible to mine a given set of data for characteristics that correlate with the behaviours observed in the set of known bads (Bolton and Hand, 2002). A *known bad* is a data indicator provided in data set with is a provable example of a fraudulent behaviour.

With that information it then becomes possible to instantiate what-if scenarios or predicate decision trees and other methods like Naive Bayesian, FOIL and RIPPER in order to effectively create a controlled means of scoring the data set and solve for a portion of the data that could be indicative of dubious behaviour.

Further to this analysis it would then be feasible according to Bolton and Hand (2002) that some form of link-based analysis could then be performed to create a network of individual customers from the data set based on linking rules. These linking rules would highlight how clients are linked through fiscal interaction and could be statistically shown to have a potential role in a fraud like scheme. Using such networks allows a statistician a means of binding risk biases on individual customers and their networks to show propensity for criminal intent and highlight the need for further investigation.

Unsupervised methods were the next evolution that the same group examined by Bolton and Hand (2001). Here they noted that charge off¹ risk and loss in fraud are highly correlated. Further, without some form of manual investigation by a human investigator at an institutions investigation unit, it would be very difficult to distinguish between the two. Thus the need became apparent for an unsupervised approach that escaped the bindings of traditional supervised approaches in detection and instead relied on mathematics and compute engines to highlight the delta between typologies.

Anomaly detection, they argue, was the natural plug that filled the void left by the supervised methods. This allowed analysts to create customer-based profiles that charted behaviour characteristics to establish norms for a given customer. Thus, using statistical distance formulae, etc. finding deviations from that norm became indicative of potential loss events and most certainly worthy of further investigation. Progressing beyond this allows institutions to garner some agility to deal with a loss event to halt replication of that behaviour in other customers thus closing the loop on a specific fraud scheme.

¹A charge-off is a debt, for example on a credit card, that is deemed unlikely to be collected by the creditor because the borrower has become substantially delinquent after a period of time.

Expanding from basic anomaly detection allows for further areas of detection. Examining the areas of text mining, Benford analysis and clustering allow for the capture of different types of loss events along the lines of temporal analysis, group behaviours and structured and unstructured analysis of the data feeds. These techniques allow for means of looking for deviations where the fraudsters is more premeditated in intent and not foiled by velocity, i.e. showing non-suspicious behaviours for a long period before entering into a short period of criminal activity before vanishing.

With these techniques, segmentation analysis and clustering become useful for dividing and grouping the customer population into sets of similar entities where observations regarding the set can show nuances on the standard deviation for detection anomalies. Furthermore, with text mining new data sources become available as the traditional transaction feed can be aggregated with log information from devices, logs and other sources. With a Bayesian approach you add updates from a trained model thus providing a bridging mechanism that evolves beyond more simplistic modeling mechanisms to push a detection engine into a more predictive system. Predictive detection technology methods allow an institution further means of stopping fraud events from occurring in real-time versus retroactive analysis.

In their work Bolton and Hand give us an academic appreciation for what traditionally was determined to be an industrial problem. They examine the mathematical methods involved in the application and posit the applications of the approaches in the field. They garner what works well in isolation versus what thrives under a bridged approach that combines both supervised and unsupervised elements. Thus their work allows us to get an appreciation for the techniques and a more complete understanding of the fundamentals behind the applications and see into the murky depths of the problems they are trying to solve for.

The methods discussed truly seem applicable to different fraud or anti-money laundering problems but at that time they did not delve into the application of artificial intelligence as a combined approach. The reason for this was that sufficient training data to highlight anomalous activity at the time of their investigation into the area

without the need for a more comprehensive model using neural networks as the problem was designed solved using less advanced techniques. As criminal schemes have evolved artificial intelligence capabilities used in an ensemble model approach have become more common-place.

Building on the work of Bolton and Hand, Otey et al. (2006) focused on the unsupervised approaches in which an unsupervised anomaly detection model is based on distances and density of events. The core behind the idea is to apply detection analytics to a local environment with a smaller data set and then productionalize it globally following a strong round of stress testing to prove reliability and robustness of the model when distributed and scaled to a large dataset.

Further, Otey focuses extensively on the composition of the data itself. The heterogeneous mixture of data necessary to capture suspicious events successfully is noted. Otey highlighted that production system datasets from the detection engines include attributes that either are continuous or categorical. Otey et al. (2006) states that categorical or nominal attributes may take on a relatively small number of values, and these values have only a partial ordering, if any at all. Whereas, continuous attributes have values that are usually numeric and have a total ordering. They argue that many techniques for outlier detection assume that the attributes in the data set are either all continuous or all categorical rather than mixed. This highlights the need in industry for augmented datasets that address not just the transactional perspective of the customer but ideally to further decorate customer records with enhanced data attributes such as external data like watch lists in order to improve the detection process.

During the process of exploring the treatment of attributes, the data is generally categorized into one of the two variable types. The allocation further implies the potential existence of *concept drift* where a target variable in the data set changes over time in unforeseen ways as the result of predictor behaviour losing accuracy over time. Another concern was also raised for both models in that density-based models tend to compute at a slower pace due to the computational intensity of the calculations and that the distance based models tend to be more linear in nature. A density-based approach to outlier detection as discussed by Breunig et al. (2000) involves a local outlier factor

(LOF) is computed for each point. Otey et al. (2006) notes that the LOF of a point is based on the ratios of the local density of the area around the point and the local densities of its neighbours. Further, the size of a neighbourhood of a point is determined by the area containing a user-supplied minimum number of points.

Otey et al. (2006) shows how these type of models scale if a model that contains a transactional or time-dependent structure is examined. Given the nature of modern data capture around financial crime, this work shows explicitly the importance of data characteristics and model performance over time. It highlights also the impact of model degradation on the effectiveness of the detection rate. The logical conclusion is that for a given model there is a limited lifetime expectancy for an acceptable rate of anomaly detection and beyond that, model degradation advances swiftly.

More recently, another examination of the state of the art by Ngai et al. (2011) highlights the difficulties in expanding the state of the art in this space. It builds on the methods discussed by Bolton and Hand (2002) and Otey et al. (2006) and observes which techniques have been adopted by industry for financial-crime related anomaly detection. Ngai et al. (2011) also provides a comprehensive overview of the banking and security fraud. Realistically though, the consequential insight from this work comes in the form of the barriers the authors faced in the attempts to enhance traditional approaches to solving for criminal intent.

Ngai et al. (2011) noted that the availability of data pertinent to fraud detection is a major stumbling block for advancing the science. This can be directly attributed to what financial institutions consider to be market risk on their carefully managed public perception. Financial institutions are always concerned about customer loyalty and what could initiate a customer churn event where they conclude all business with the bank so this impacts the ability of compliance departments to legitimately exit a client. Further, the perception of criminal organizations on what institution represents the weakest of the herd is sometimes measured by the deterrence mechanisms each bank utilizes and which one is lagging in terms of technology.

Thus, industry is highly resistant to sharing data with academia and has stymied the efforts to enhance and improve existing methods and models dedicated to this space. Ngai et al. (2011) is a departure from work like Phua et al. (2005) where the focus is on the core themes of supervised and unsupervised approaches. Extrapolating from the works of previous research into the area they posit that in the case of supervised learning, it is suggested to avoid using rule-based predicate trees or nearest neighbour-based matching mechanics unless the data is seeded with a large number of positive cases.

Further, a degree of concern was raised in methods used in the unsupervised space. Typically the implications of utilizing model-based approaches mean time has to be dedicated to training the model, time which given the potential lifetime of a model may or may not provide a significant return in investment. Instead, the idea around utilization of a two stage classifier system is suggested: One phase, which scores based on randomly drawn positive and negative entities within the dataset and a second phase then average scores the entity and its neighbours to reduce false positives. The first approach really is a sampling approach utilizing seeds to bias the score of a particular population which it observes over the complete segment. The second looks at the individual and the insights the system already has to calculate a propensity likelihood that the individual is fraudulent in comparison to a much small cluster of statistically similar individuals.

Extrapolating from this work, we can gather that models are consuming either from an I/O (data management/movement) perspective or on the computational side in the form of analytical processing. This raises questions on where to better focus for a resolution, either on the data side through better data scalability and a reduction in the functional movement of data elements or to improve accessibility to the data space through some form of in-memory capability that is capable of distributing the computational processes.

Garnering from these insights collectively, the concern of scalability, performance and degradation of modelling based techniques applied to this space is raised. One notes only one specific absence within the cycle that is the application of stepwise

regression within the space. This is discussed at length by Foster and Stine (2004) where they discuss the utilization of the technique to solve for a specific problem in industry. This paper shows the power of applying analytics to the complete life cycle of an industrial issue and seeing what came to light when utilizing models over the lifetime of the data.

What comes to light is evidence that shows that, had statistical techniques been applied at the start of the process, there was a correlation which showed bankruptcy could be predicted. This really enforces the most important outcome of both spaces the need for a scalable means that can predict loss events without the need for multiple instances of a loss type occurring. Seeded bads and rules utilized in supervision can in isolation protect institutions. However it is our opinion that only with an ensemble approach mix of both methodologies can we successfully ensure a modicum of security across the dataset under examination and have the certainty that predictive measures with adequate scrutiny can indeed model and show potential fraud anomalies.

4.3.1.1 Link Analysis

Diving deeper into the realm of link analysis we can trace the origin of the techniques application in this space to work conducted by Goldberg and Senator (1995). That work examines the application of link-based analytics in the area of entity resolution. This is effectively where separate silos of data are compared using known attribute links to score the likelihood of similarity between two distinct entities. The classic example here is the determination in financial institutions where it is statistically likely that an individual named in one account is the same individual named in another.

The primary application in Goldberg's work is applied to FinCEN (Financial Crime Enforcement Net) data which focuses around AML data and entity resolution for the purposes of identifying suspect money launderers and their activities in a series of different institutions. By being able to create link-based scores on the networks themselves Goldberg is quickly able to posit a high correlation in determining that a criminal is engaging in illicit activity in a number of banks.

Continuing in the same vein, Goldberg continued to apply link analysis years later (Goldberg and Wong, 1998), again on FinCEN data. The variance between this and the original work was the use of temporal characteristics on the link analysis network and applying their transactional reduce technique for network score refinement. This is interesting as it shows score behaviours adjusting over time as new node entities join, change or leave the link network.

Kubica et al. (2002) take the method and push it into the predictive space. Here they start adding predictive behaviour to the link analysis when determining the next likely link that will be added to the network. Extrapolating from this they are effectively stating that they are able to statistically predict within their experiments that certain data elements within that dataset will, over time, attain a link to the network. This effectively becomes a function of transactional behaviour that can be validated.

Where this specifically can be interesting is through observation of initial characteristics of the dataset in the purposes of fraud detection. Assuming a known set of behaviours or known fraud schemes seeded within the model it should then, with predictive capabilities, be feasible to apply this to extended data sets to posit potential fraudsters within the data.

With their work Kubica create fuzzy or soft links to show the relationships between entities but it was not really until the work of Xu and Chen (2003) that this work was combined with path algorithms to truly evolve the technique into a fraud detection method.

In their work Xu and Chen utilized the shortest path algorithm to find the minimum distances between two nodes based on given criteria. This allowed the model to isolate and focus on the key player in a fraud ring over time. This enabled them to quickly identify contexts where a group was working together to purposely defraud an institution and score the network thus. Using a path algorithm allowed them to vary links and nodes in such a way that the central individual in that crime ring was displayed in a hub and spoke type evolution of the link analysis graph of the dataset.

Following up on this further improvements to the method were made (Xu and Chen, 2005) by contrasting three core approaches to the link analysis method. They divide these into generations where the first generation effectively was a manually-based method of link analysis through the dataset itself.

They posit the second generation as being a graph-based application that complements the first generation such that a network visualization is formed.

In the third generation of the approach they add relational analysis to the network. This is based on node-based concepts of degree, betweenness and closeness. This allowed them to look at the dataset and create node clusters and adapt key decision scores around where a given node sat in the relationship tree and if it exhibited one of the characteristics noted above. This creates link biases to prioritize a link to a node given multiple links to the same node in the tree.

An example of this would be to consider all the actors interacting in a new home purchase scenario. Actors would include buyers, sellers, relationship managers, bank managers, lawyers, developers, realtors as an example. Several of these actors would have repeat interactions over the course of time on different homes. Others would be far more infrequent actors to the chain as the purchase of a new home should be an infrequent event. Thus, we can consider the links between actors that often pursue this type of scenario like realtors, bank personnel and developers to be close but their relationship to buyers and sellers more distant. This would be reflective of the third generation of the network.

They then created a series of attributes through the use of a position analysis-based approach with this generation which allowed them to score the position of the node in the network. The difference to the previous approach is that if a node had a number of different links each link was scored the same but the subset is scored in importance to other nodes and their subsequent analysis links in the network.

The natural evolution from both of these techniques was a hierarchical clustering of the link analysis network. This meant that, for the relational approach, clustering created subgroups or subnets on the original network designed to be of statistical interest. Clustering allows the user to determine similarity between nodes, which is an enhanced form of entity resolution. This work thus imparts a very important division of application within the link analysis.

Depending on the context of observation you can utilize either Xu and Chen (2003) or Xu and Chen (2005) but Xu and Chen (2003) appears better for determining fraud type behaviours while the Xu and Chen (2005) appears better in the context of entity resolution or perhaps in noting statistical similarity between node clusters.

Other work around the same time includes Getoor and Diehl (2005) which looks at other interesting techniques that could be applied in this domain. This examines concepts like link-based object ranking in order to display centrality for a given network. This also allows for the assignment of scores for entities and network solely based on data and linkages but does so at a more granular level of detail.

They next discuss object clustering (group detection) and object identification (entity resolution) like the previously discussed work of Xu and Chen, where clusters of data of similar characteristics can be determined and visualized in the dataset. Filtering approaches for the network are discussed where subgraph discovery, using AGM (Apriori based algorithm¹) to induces in the data as well as applying some other form of group classification mechanism.

This is an extension of graphing-based methods to classify a network and further reduce it for various means. Significant research into the generative models for a range of graph and dependency types have been researched around the social network analysis theme. Getoor and Diehl (2005) notes specifically that for directed graphs with a single object and link type, there are several major classes of random graph distributions. Primarily, it appears to be most useful in reducing false positive matches within the

¹Apriori is an algorithm for looks at the frequency of items for set mining and association rule learning over transactional databases.

network and refining the overall network in terms of fraud applications to determine a high propensity for a fraud scheme match.

All these techniques and the evolution of the approach into graphical visualizations made it very evident that there was a big problem with super clusters. This is where every entity has some form of link between them and determining suspicious activity becomes highly improbable. Thus, work like Yang et al. (2006) becomes interesting to determine how we can refine a super cluster into usable groups within it.

Here they apply a Spring Embedder Algorithm (Eades, 1984) to the network to improve the aesthetic qualities of the graph. This is a method of reducing network complexity based purely on data structure without the necessity of problem domain knowledge and is also known as force-directed graph drawing. The procedure effectively creates a subgroup of the underlying network to show simpler networks for high score sub-graphs. By applying domain knowledge to further refine this sub network we can hunt more effectively in this network graph window for suspect entities.

They also explore the concept of changing the overlay of the node layout on the graph for optimizing detection of suspicious entities by the human investigator. To do so they use different views on the graphical network diagram like fish eye and fractal views. This allows the investigator to enlarge regions of interest within the network for study and exploration while diminishing the rest of the network for the scope of view. This technique allows the investigator to improve clarity of the network and the cleanliness of the link analysis outcome.

In most investigation units trying to solve for financial crime problems they seek to have a marriage between techniques like Xu and Chen (2005) and Yang et al. (2006) in order to ensure that investigators are examining higher value networks that are comprehensible, rather than looking at noisy networks that are more like super clusters.

4.3.1.2 Text Analytics

Delving into the world of text analytics allows us a means of data mining information sources for curious patterns. Patterns that are not immediately apparent to the human

eye but come to light through analytical investigation and the use of technology are particularly relevant in the realm of fraud detection.

When we examine earlier work in this area like Rosset et al. (1999) where they look to determine new means of optimizing and operationalizing rules in the telecommunication context, we see that text mining is a method of search that consumes and analyses data for user determined characteristics and acts within a specific binding scope.

In contrast to the normal use of the Greedy Algorithm where the rule selection always takes the best immediate, or local, solution while finding an answer (Black, 2005), in the work of Rosset et al. (1999) we see the use of non-greedy rule selection where optimization efforts are applied to create a more robust selection process. This theoretically allows for more in-depth analysis of text streams to allow for classification prioritization to be applied on taxonomies that map to financial crime patterns. What they effectively do is stratify rule selection for different tiers (prioritizations) such that they have high specificity and sensitivity at the customer (case) level. Further, they then ensure that they have high coverage of true fraud alerts at the behaviour level. Thus by creating strata for classification of rules and data patterns they are in a position to compute accuracy and coverage indices to serve as benchmarks for the effectiveness of the fraud rule detection strategy.

Bolton and Hand (2001) utilize a similar technique for the purposes of segmentation of the customer base from the transactional information. With these separate, yet intrinsically linked, streams due to the commonality of data elements in both, Bolton and Hand are rapidly able to create clusters based on behavioural characteristics of both the customer set and transactions.

With these clusters they then seek out outlier transactions and compare them against the customer cluster for risk score evaluation, thereby stratifying the customer base into risk clusters such that they can be targeting with specific models and rules for fraud identification. A good example to consider is the concept of employment type risk scoring. One could argue that some professions are of higher risk than others and then

further evolve the taxonomy with other indicators to improve the risk clustering further. Example professions would include professions like realtors, lawyers and accountants in high risk jurisdictions.

By using analytics in this fashion against the data sets Bolton and Hand are able to ensure better scalability in applying models to what are traditionally static and dynamic data sets, while ensuring justifiable outcomes which are indicative of loss at the nexus of both streams.

Kim et al. (2003) examine and propose an expansion to the breadth of the search capabilities in order to expand upon the customer profiling abilities by increasing the number of dimensions considered. They do so from an algorithmic perspective by utilizing a Support Vector Machine (SVM) ensemble. SVM is defined as a supervised machine learning algorithm used for classification and regression analysis. It works well in the financial crime context due to the quantity of structured labelled data available to the financial entity trying to find criminal behaviour patterns in their data.

Extrapolating from this paper we can envision this technique being utilized to augment traditional static perspectives of the customer or fraud schema with characteristics from a modelling perspective. This would show annotative clustering of that customer segment and where this customer sits in that regard from a risk perspective. This would be of significant value for investigative purposes and is a unique contribution of the method.

The theme of using stepwise regression in some form to augment our understanding of customer behaviours is further explored by Foster and Stine (2001). In this work, using data mining techniques, the authors attempt to create a predictive model for bankruptcy. In this case we are interested as bankruptcy predictors can also be similar from a risk indicator perspective to fraud, as many fraud schemes follow a similar pattern. An example of frauds related to this area would be cheque kiting. Cheque kiting is a form of fraud, involving taking advantage of the float to make use of non-existent funds in a checking or other bank account.

One area Foster and Stine (2001) highlights though is that validation of the model given the dynamic nature of the data can be difficult. Work like Fan (2004) explores this area in depth. It posits that in the case of unknown data sufficiency and concept drift, the use of a cross validation decision tree supervised method helps to reduce the false positive rate for detection.

As both temporal data elements and fraudulent behaviours are in a constant state of flux, mining the temporal data becomes burdensome and computationally difficult. As such, Fan (2004) promotes a means for pattern recognition that utilizes the *cross validation decision tree algorithm* to reduce dependencies of past events and to minimize the impact of stale data on the scoring mechanics.

What this effectively means is that the method continuously evaluates the effectiveness of a particular risk bias on a score. Its logic determines that if the bias is weighted by stale data elements, it biases that rule towards lower prioritization in the isolation of fraud events.

4.3.1.3 Neural Networks

The use of different aspects of artificial intelligence methods has also penetrated the domain of fraud detection. Ghosh and Reilly (1994) present an introductory use of the neural networks applied to fraud where they utilize the approach for credit card contexts. Their approach is to utilize a more supervised approach using P-RCE neural networks. A P-RCE neural network is a radial basis function which was developed for use in pattern recognition systems. In modelling, a radial basis function network is an artificial neural network that uses the functions as activation functions for the network. The output of the network is a linear combination of radial basis functions of the inputs and neuron parameters. Radial basis function networks have many uses beyond the pattern recognition use case discussed by Ghosh and Reilly (1994), including function approximation, time series prediction, classification, all of which are recurring areas of interest within the financial crime domain.

The authors describe their P-RCE neural networks as being a three-layer, feed forward network following the established pattern of other radial basis functions. They distinguish it from its peers through the fact that it only requires two training passes through a data set rather than other approaches which use significantly more training. They apply the neural network to NRI (non-receipt of issue) fraud which is a specific fraud scheme targeting credit cards. They also utilize it in risk assessment and mortgage assessment issues on determining potential loss patterns. The paper shows a 20-40% loss reduction from fraud losses and operational efficiency gains comparatively to other traditional approaches.

Further work using neural networks in card fraud detection was investigated by Brause et al. (1999). Here they evaluated success criteria for a given model and the likely propensity of fraud occurrences across the breadth of financial activity. As the likelihood for an event to occur is statistically so low in their opinion, it garners the need for almost perfect capabilities to detect a potential loss event on a given instance.

Their approach is to use a more context-based approach and to use a 4-layer neural network. This is to allow them to incorporate domain knowledge in order to instantiate *do not care* variables from transactional data and use the neural networks to then seek out patterns in the transactions from the tweaked data set.

Syeda et al. (2002) seek granularity in the approach and their work focuses on a more granular type of neural network. Here the primary driver is based on building better efficiencies based on insights from the training data but the method proposed also pushes computational complexity to a 5-layer tactic which is differently from previously discussed approaches like Ghosh and Reilly (1994).

With parallel processing for lowering training time they are able to garner some operational efficiencies from the technique. They focus their efforts at combining transactional data from various channels in order to get better prediction outcomes as well as to reduce the probability of model training induced errors. They succeed in this goal and as a result are able to have a considerable impact on fraud detection rates within the data.

Training and the data it requires is a popular concern as shown by Barse et al. (2003). The delta here between their approach and that of their predecessors is to utilize synthetic data to train the neural network rather than to utilize traditional seeded data sets. The synthetic data in this case is generated on the audit trail or authentication data from the cards transactions, rather than just looking at posted transactions. This means far more potential fraud detection as the training capability broadens the event focus away from traditional fraud characteristics and into areas which would show tester authentications and other interesting validators.

Srivastava et al. (2008) attempts to broaden the scope of neural networks within the fraud domain. Here they do so by utilizing a hidden Markov model on the credit card training data associated with the potential loss events. As per the norm the network needs to be first trained with normalized behaviour for a given segment of cardholders.

They then use this knowledge and apply it in deciding the value of observation symbols and some initial parameters for the model. The model therefore is able to discern and predict a potential loss event based on deviations from what it considers to be normalized behaviour for a given card-holder. If the behaviour is not consistent with the observations it has made, the model logic dictates that this is evidence of fraud and escalates the risk correlation score accordingly. The authors claim to have high scalability but due to the nature of human transaction profiles for any card portfolio it also could incur massive levels of false positive matches as it can not predict human behaviour with complete accuracy.

Contrasting the methods discussed above we can note how the current state of the art evolved to use more training data sets in order to fine tune detection models. The breadth of the data also grew to include non-traditional elements like Barse et al. (2003) introduces. Each tweak to the neural networking approach successfully improved fraud loss avoidance and allows financial institutions a more statistically driven mechanism to react faster to fraud schemes and thus become less reactionary in nature.

4.3.1.4 Bayesian Networks

In probabilistic graphical models, Bayesian network theorems help to explicitly express the cause-and-effect among variables in graphical structure and its applications. This has led to significant research attention to the technique. In the realm of financial crime, the Bayesian network model can be used to discover complex hidden relationships between entities. However, it has been argued in research papers that a Bayesian network can be counter-productive in both training time as well as accuracy in prediction if not tailored for specific data structure as a majority of the Bayesian models are constructed to fit the database.

Ezawa et al. (1996) created the goal-oriented K2 algorithm with the purpose of predicting collections-related risks¹ in a telecommunication risk dataset based on APRI, a supervised Bayesian network model. The proposed systems modify the Bayesian network by introducing a class node for prediction, which provides a tailored model specific for this problem. In short, each variable expects to have effect on the class variable and aggregation of variables would further constitute significant impact on classification accuracy.

As part of the empirical outcome, the group demonstrated that the proposed algorithm can be effective to more accurately solve unbalanced binary classification questions. Thereby providing lift when applied to collection problems in telecommunication risk data. This risk data is what is traditionally used to find payment concerns within the customer base and provides direct links to fraudulent use cases.

Maes et al. (1993) applied a Bayesian type network model in a credit card fraud experiment aiming to characterize and to identify strength and weakness with STAGE Bayesian network approach. By comparing with the neural network, the Bayesian network training time is relatively shorter, but longer when applying to new problem domains outside of the provided example. True positive rate is higher compared to the neural network, but they noted that the false positive rate is also higher. As discussed by the group, a further enhanced Bayesian network approach to financial crime

¹Collections is a process that financial institutions utilize to recoup assets from clients that are in a deviant account status for financial or fraudulent reasons.

is suggested in which the necessity of incorporating both continuous variables and a structure-learning mechanism based on dependency analysis should be considered.

Borsuk et al. (2004) proposed the concept of applying Bayesian network to visualize and understand relationships that are often ignored or overlooked. Simplicity is among the key factors that were discussed. Based on the heuristic knowledge, the complexity of the Bayesian network can be reduced by trimming the number of nodes. Subsequently, a fictitious network mirroring the real-world scenario can be created. By constructing the simplified version of the underlying scenario, the requisite time for integrating with other models can be reduced. This can be an effective strategy to winnow down complex fraud schemes into more consumable elements and allow for validation using other techniques.

Viaene et al. (2002) demonstrated the use of AdaBoosted naive Bayes scoring which provides the advantage of boosting and the flexibility of heuristic knowledge integration for the model. AdaBoosted naive Bayes scoring combines the use of naive Bayesian classifiers with tree structures in order to achieve significantly lower average error than both the naive Bayesian classifier and boosting the naive Bayesian classifier itself. Ting and Zheng (2003) subsequently develop a method of successfully applying the boosting technique to naive Bayesian classification. The weight for each elementary node can be aggregated and used to create the risk scores which further improve the pattern recognition found in the positive data. The model can be used to support operational investigative discovery by assisting the domain experts performing sensitivity analysis and running what-if scenarios.

Bayesian networks have been used in various financial crime detection frameworks to observe previously indeterminable patterns like unknown cause-and-effect relationships using probabilistic modelling. In one attempt to depict advantages of Bayesian networks and graphical models, Kubat et al. (1997) argued that in the case of rarity of positive data, the accuracy of a Bayesian network model might be impacted significantly. This is a challenge faced in compliance-related financial crime as an institution often obtains few true positives to compare against due to regulatory restrictions on

responding to suspicious transaction reports. To date regulators decline to provide information back to financial institutions on whether the regulatory report filed led to a criminal charge or not.

4.3.1.5 Positive and Negative Match

In empirical science, systems and models are built based on the results of experiment. Traditionally, both supervised and unsupervised algorithms are developed with the presumption both positive and negative data are distributed with relative similarity in population where the probability of sampling positive data is comparable to the negative. However, in the financial crime domain, the positive data often contributes approximately 0.1% to the overall population. In such an extreme environment where positive data is more infrequent than negative data, many of the established and mature modeling techniques are being challenged in terms of whether these models can properly adapt to the imbalanced structure of data.

In addition, in the real-world, imperfection of the data creates additional burdens to achieve theoretical expectation in both performances and accuracy for various models. It is important to consider the data environment prior to deciding which types of model to apply and ensuring high adaptivity of that model to the target specific domain problem. Various research groups have contributed and focused on the topics of sampling and models enhancement while factoring rarity of positive data as a main challenge in the process of data analytics and predictive analysis.

A framework summary was published by Weiss (2004) to illustrate several methodology issues, such as improper evaluation metrics, lack of data, data fragmentation, inductive bias and noise as the toughest challenges to surmount in the application of data analytics. Weiss also proposes several methods for addressing rarity such as improving evaluation metrics, applying non-greedy search technique, incorporating heuristic knowledge, etc. They define rarity as ideally being defined with respect to the underlying distribution for a given dataset. The framework provides an overall picture of techniques that could be applicable. In financial crime, the institution faces challenges as the result of absolute rarity and imbalanced weighting heavily biased to the rarity.

The relevance of this becomes important when one starts considering the daunting task of identifying rare pieces of data which could point to elements of terrorist financing or client risk data attributes that previously were not mined by the analysts responsible for compliance.

As mentioned previously, the imprecise environment of a real-world scenario results in challenges for operational conditions which further degrades the performance of the model. Provost and Fawcett (2001) present a robust classification model - *Receiver Operating Characteristic convex hull* (ROCCH) that seeks to minimize the uncertainty caused by the imprecise real-world environment as well as improving the efficiency, scalability, and ease of management. The ROCCH is a hybrid approach by integrating decision analysis with ROC analysis and further optimizing the AUC (*area under curve/C-Statistic*) by selecting the best fit classifiers.

Several approaches have been discussed and developed over the years to address the imbalanced dataset in various scenario. Several research groups published approaches utilizing over-sampling, under-sampling, one-side-sampling, etc. One model that has attracted significant discussion is SMOTE, Synthetic Minority Over-Sampling Technique discussed by Bowyer et al. (2011). In short, the group demonstrates SMOTE can be used to improve the AUC in ROC space by forming a new minority group as the result of interpolating between minority classes.

The new data samples are synthetically fabricated by applying noise to the minority group hence improving the accuracy by avoiding the over fitting normally as the outcome of traditional oversampling techniques.

Building on the SMOTE algorithm, Batista et al. (2005) proposed that by integrating SMOTE with Tomek link¹, or ENN², the overall accuracy can be improved based on multiple empirical outcomes where these techniques score the highest AUC among

¹Tomek links remove unwanted overlap between classes where majority class links are removed until all minimally distanced nearest neighbour pairs are of the same class.

²Wilson's Edited Nearest Neighbour Rule which classifies an input sample by assigning it to the class of the majority of its k closest prototypes

other under-sampling and oversampling techniques. One note worth mentioning is that the experimental data elements are selected in order to capture imbalanced datasets.

A fresh view on the problem in Liu et al. (2009) discusses an under sampling technique. To compensate the inaccuracy caused by the imbalanced dataset, the group proposes two approaches, EssayEnsemble and BalanceCascade. Essentially, to counter the exclusion of potentially useful data by under sampling, the idea is to randomly break down the negative data into smaller sizes and construct classifiers with the positive data. Liu defines that EasyEnsemble samples several subsets from the majority class, trains a learner using each of them, and combines the outputs of those learners. In contrast, the BalanceCascade removes the negative data used once classified and repeats the training process.

In the financial crime space, rarity and imprecise environment have been deemed to be of the most crucial and obstructive challenges in applying advanced data analytics. Training data size, class prior, cost of errors, and placement of decision are key criteria in phases of system variables determination and model selection, as pointed out by Breiman (1996). The arbitrage characteristic of ROCCH model can be used to reduce the impact when interpolating new classifiers as well as honing the performance efficiency to reduce the operation management involvement. SMOTE and its child methods, SMOTE-KNN and SMOTE-Tomek, and under sampling ensemble methods, EssayEnsemble and BalanceCascade, show that they can successfully provide detection improvements in financial crime systems.

4.4 Financial Crime Model Examples

As we have already ascertained there are numerous types of detection problems within the financial crime domain. To crystallize the value of document driven specifications for this space it is important to understand some of the complexity associated with the space by examining some examples which attempt to solve (through code) several areas of concern.

4.4.1 Name Recognition in Financial Crime Applications

Name matching is an important part of anti-money laundering activity where it is incumbent on financial institutions to examine their client bases to identify potential links to terrorism, human trafficking, drugs etc. There are a series of generated data sets that several regulators generate to target specific individuals around the globe and it is subsequently illegal to bank these individuals or their associates, such as the sample data used in Figure 4.1 for illustrating the approach. Any banking entity as a result is forced to scan their client lists looking for said individuals, links to them or places where they potential are acting as ultimate beneficial owners of accounts.

The following method using the SAS modelling language (Barr and Goodnight, 1976) takes advantage of two core procedures to conduct the name matching function (Russell, 2018). In this case they simplify the problem for illustration purposes against a series of simple data elements but this approach scales to solve for fuzzy matching in larger more appropriate data sets also.

The first function, *SOUNDEX* in Figure 4.2, is a realization of the algorithm used within the SAS language to identify phonetic elements in English names. Apache (2018) is a project that provides similar algorithms to *SOUNDEX* but these algorithms provide support for other languages phonetically. This is a complex issue as some languages are more tonal and the impact on the algorithms can be severe for phonetic matching.

The second major function, *COMPGED*, returns the generalized edit distance between two strings according to Barr and Goodnight (1976). This code looks at the Levenshtein edit distance which examines strings and provides a score on how dissimilar they are to each other. It operates by breaking each string into its specific character components and examining the number of operations required to transform the second string to match the first.

The example above provides a simple perspective on the name matching problem using names like the ones observed in Figure 4.1. The increased complexity occurs when you consider the multidimensionality of financial data and start to include other

Figure 4.1: Name Recognition Example

```
/* Sample data that contains names and the class being taken */
/
data class;
input fname : $12. lname $ class : $9.;
datalines;
Jon Smith Math
John Smith Math
Johnny Smith Math
James Smith Math
Will Miller Chemistry
Willy Miller Chemistry
Willie Miller Chemistry
Bonny Davis Gym
Milly Wilson Biology
;

/* Data set that contains the name and the grade for the class
*/
data grade;
input fname : $12. lname $ grade $;
datalines;
Johnathan Smith A
William Miller B
;
```

Figure 4.2: Name Recognition SOUNDEX example

```
data c;
set class;
tmp1=soundex(fname);
do i=1 to nobs;
set grade(rename=(fname=fname2)) point=i nobs=nobs;
tmp2=soundex(fname2);
dif=compge(tmp1,tmp2);
if dif<=50 then do;
possible_match='Yes';
drop i tmp1 tmp2 fname2;
output;
end;
end;
run;
proc print; run;
```

pertinent fields like phone numbers, addresses, known associates, countries, email addresses etc. Further, there are other considerations surrounding language and character sets to further complicate matters. This is exactly why a comprehensive perspective is necessary in order to ensure that all fields are being consumed and scored appropriately and where formal methods specifically can add a lot of value.

4.4.2 A Neural Networking Fraud Detection Example

Earlier in the chapter, Section 4.3.1.3, we introduced neural networks as a central element for techniques to solve for financial crime. Neural networks are used to detect a variety of complex financial crime patterns as evidenced by Phua et al. (2005). The method is most useful when dealing with large volumes of data and a diverse population where multiple observations are necessary to identify suspicious behaviours.

The ability to identify these behaviours requires information where true positives indicate the presence of the crime of interest. These indicators included information like common phone numbers, emails etc. used in multiple crimes where a scheme was detected. Often however these schemes are determined after the fact hence the need to have a system in place which helps financial crime analysts identify the crime more

Figure 4.3: Neural Networks Example Step 1

```
/* Step 1. */
libname mylasr sasiola host="fraud.example.com" port=10010 tag
    ='hps';

%let base = https://www.kaggle.com/ntnu-testimon/paysim1;
data mylasr.fraudbase;
infile "&base/fraudbase/fraudbase.data" device=url dsd dlm=', '
    ;
input step type amount nameOrig oldbalanceOrig newbalanceOrig
    nameDest oldbalanceDest newbalanceDest isFraud
    isFlaggedFraud Cap_Avg Cap_Long Cap_Total Class;
run;

proc imstat;
```

swiftly and halt the loss of capital from the accounts at risk. The training data sets used generally provide over a years worth of data inclusive of both good and bad actors.

To illustrate the method the following example illustrates the training of a model using a publicly available dataset based on the work detailed by Lopez-Rojas et al. (2016). The example is modelled in this case for a fraud context based on the dataset created by Lopez-Rojas et al. (2016) using SAS modelling code (Barr and Goodnight, 1976). This approach targets the SAS LASR facility for in-memory analytics across multiple processors. The idea behind that is because fraud is a time-dependent problem the faster we can have results back the better a position we are in to counteract the affect of a specific scam.

This code examines the data in several phases. The first phase, Figure 4.3, is a consumption phase where the dataset is consumed into the analytical environment and the variables are set up for usage. The data can be consumed from a variety of places: data stores, a URL, flat files etc.

The next phase, Figure 4.4, is to pre-train several 'shallow' neural networks starting from different points in the data set. This helps avoid creating a neural network that is

Figure 4.4: Neural Networks Example Step 2

```

/* Step 2.
table mylasr.train;
where part <= .75;
NEURAL class/ seed=12345 details temptable
/*input */ input=(make--cap_total) std=std
/*target*/ targetact=softmax targetcomb=linear error=entropy
    nominal=class
/*hidden*/ hiddens=(10) act=(logistic) combine=(linear)
/*prelim*/ numtries=5 maxiter=10 tech=congra
/*NLOP */ maxfunc=1000000 linesearch=2 fconv=1e-4 lower=-20
    upper=20;
run;

```

ineffective due to poor initial values seeded within the dataset. An ineffective network reduces the efficacy of fraud detection and negates any potential lift overall.

Building on this we next select the best neural network from the pre-trained neural networks we just built, Figure 4.5. Using this neural network as a foundation we attempt to train a much deeper neural network as the final model.

The *ASSESS* option specifies to add the predicted probabilities calculated to the scored data for all the levels of the nominal target variable. In this example, Figure 4.6, two levels are created because the variable named class has two values, 0 or 1. The scored data are stored in a temporary table.

Next we perform model assessment using the scoring result in order to gauge how effective the model will be at finding fraud, Figure 4.7. The probabilities of all levels are in output, but only the probabilities of the event level are necessary for analysis. The *WHERE* clause is used to select the rows with event level only from the output. The *STRIP* function is applied to remove the blanks in character variable `_NN_Level_`.

One of the most important parts of being able to understand whether or not a specific model provides the necessary lift is the use of model validation. Often the most popular

Figure 4.5: Neural Networks Example Step 3

```

/* Step 3. */
table mylasr.train;
where part <= .75;
NEURAL class/ seed=12345 details temptable
resume LASRANN=mylasr.&_templast_
/*input */ input=(make--cap_total) std=std
/*target*/ targetact=softmax targetcomb=linear error=entropy
        nominal=class
/*hidden*/ hiddens=(10) act=(logistic) combine=(linear)
/*train */ tech=congra maxiter=50
/*NLOP */ maxfunc=1000000 linesearch=2 fconv=1e-4 lower=-20
        upper=20;
run;

```

Figure 4.6: Neural Networks Example Step 4

```

/* Step 4. */

table mylasr.valid;
where part > .75;
NEURAL class / lasrANN=mylasr.&_templast_
input = (make--cap_total) nominal=class temptable assess vars
        = (class);
run;

```

Figure 4.7: Neural Networks Example Step 5

```

/* Step 5. */
table mylasr.&_templast_;
where strip(_NN_Level_) eq '1';
assess _NN_P_/ y = class event = '1' nbins = 20 step = 0.05;
ods output ROCInfo=work.rocdata LiftInfo=work.liftdata;
run;
quit;
proc lasr term port=&myport;
run;

```

Figure 4.8: Neural Networks Example Step 6

```
/* Step 6. */
proc sgplot data=liftdata;
series x = depth y = Cumlift / markers markerattrs=(symbol=
  circlefilled);
series x = depth y = CumliftBest;
yaxis label='L' grid;
run;
data endpoint;
Sensitivity=0;
Specificity=1;
run;
data rocdata;
set rocdata endpoint;
One_minus_Specificity = 1 - Specificity;
run;
proc sort data=rocdata;
by One_minus_Specificity;
run;
proc sgplot data=rocdata;
series x = one_minus_Specificity y = Sensitivity / lineattrs=(
  color=blue);
series x = one_minus_Specificity y = one_minus_Specificity /
  lineattrs=(color=black);
yaxis grid;
run;
quit;
```

validations are the use of lift and ROC (Receiver Operating Characteristic) curves. In this use case we would derive these from the result of ASSESS statement used above. Using the visualizations generated, by 4.8, we would be quickly able to assess if our model had any tangible impact in finding fraud within the dataset by comparing its results to the known frauds seeded in a test data set.

Given that this represents just a neural networking functional approach to looking at a financial crime it is important to note that the effective complexity grows as you start to consider more model features and more comprehensive data sets describing customer interactions. As these models grow in complexity the ability to have a com-

prehensible asset that ensures that a given model meets its intended design becomes more necessary. Some of this void is filled by the use of developer driven commentary but often when a model is put into production there is no guarantee that the description is effective enough to explain nuance to a regulator or peer modeller.

Formal methods represents a significant improvement on basic comments for explaining model outputs as it provides a more effective means of highlighting all aspects of the program's intended definitions. Given the stringent regulations that regulators have on the banking sector and the severe financial penalties for non-compliance we argue that enabling the sector to have more of a document driven design approach would be a benefit to all parties involved. This holds true only if the documentation is maintained and is reflective of the software as it is augmented and built upon.

4.5 Analysis

Financial crime applications within the anti-money laundering domain face a very difficult task in satisfying regulators that they provide adequate coverage in deterring criminal activity seeking to take advantage of a financial institution. We argue that the answer to this concern is to utilize the TFM to provide precise and readable documentation suitable for both machine computation and human review.

When looking at linking formal methods to the world of financial crime specifications there are a number of concerns that immediately leap out. One immediate concern is the management of Bayesian predictive modelling space and neural networks. We can see a correlation on the management of training data to that of tabular expression constructs as each record over time should generate a boolean link in terms of a *if-then-else* style of construct. However the risk comes in mapping the model grouping mechanism into clusters and how to map that network into a form that can be easily checked. Neither Ezawa et al. (1996), Maes et al. (1993) nor Borsuk et al. (2004) provide us with a bridging mechanism to do so. Potential biasing methods could be inferred from Viaene et al. (2002) given the utilization of weights and scores. Regardless, for the scope of this research we will exclude this area as it is too far a stretch goal from the specification logic described by Janicki and Khedri (2001).

Positive-Negative match is a clearer mapping as we can see native matching in terms of predicate statements between values held in an array akin to a tabular expression format. Due to the relative comparative ease offered by a parsing function in the kernel element offered by Bane (2008), translating fuzzing matching style mechanisms to an array of values should be feasible. Further, the predicates map cleanly to proving strategies and should for the most part escape overwrought complexity.

Text analytics also represents a challenge but less of a challenge in comparison to Bayesian or Neural Networks. This is predominantly because text analytics is effectively an advanced pattern recognition system for unstructured data sets like Rosset et al. (1999) discussed. An array of indicators can be assessed from a storage array held in tabular expression format. The specification mapped should be able to assess incoming data streams against that specified entity list and evaluate down to predicate checking solvable by an automated mechanized mathematical tool proving strategy.

Link analysis-type specification trees like Goldberg and Senator (1995) or Goldberg and Wong (1998) would appear to have a translation path to tabular expression formats that works well and should be easily understood. Assuming that we can create logically sound specifications of the link definition structure, it should be feasible that we could create expression trees of suitable completeness and disjointness to represent the specifications.

While it is not optimal that we are unable to consider all aspects of financial crime detection methods, we believe we have enough cross over to study the application of formal methods to this unique form of critical system. Of the above treated areas each bring unique approaches to describing a financial crime model.

It is clear though that the trace function method offers a lot of advantages when considering programming models based around risk scoring attributes. This is because this method offers capabilities above and beyond those of the other methods discussed. Primarily, the logic of the left-right sequence reading of the traces themselves allow for ease of comprehension with the associated module Parnas (1992). Further, with the

addition of events the method potentially allows for tighter alignment with the evolution of risk scores over time but this would need further testing to validate. Another potential win would be the effective functional library offered by the method which allows the developer access to several descriptive functions that empower them for more compact descriptions (Bane et al., 2004), however if these have be utilized without tooling this negates the impact severely.

Ultimately the biggest win by potentially using the method could be that it might help reduce the possibility of errors within the code being deployed to detect anti-money laundering. This would be a twofold success in that it satisfies the constraints on institutions placed on them by national and international regulators. Second, it provides an accurate depiction of all aspects of the deployed code thereby ensuring ease of adaptation to suit new regulations and to counter new criminal schemes. When one considers the breadth of the legalities that compose a regional wide directive like EU (2012) it is important to note the diligence required to translate the directive into policy for a given institution. Further, the translation of policy into operations is equally complex. Hence the need for accuracy at all levels of the institution becomes paramount.

Another consideration is the comprehensive nature of being able to prove compliance. Technology provides us with a wealth of mechanisms to evaluate and audit data and this is especially true within the financial services sector where such audits are a strict part of their review process. Compliance concerns result in severe financial penalties and human driven review cycles are slow, tedious and also prone to error. The TFM could provides us with an ancillary mechanism to inspect, audit and evaluate the coverage of the financial crime system by using technology to verify the completeness of the specifications and could short cut the time necessary to present results back to regulators and costs involved for banks. If this hypothesis holds true, we cannot underscore enough the value of both time and cost savings to financial institutions and in particular the IT departments responsible for maintaining financial crime detection systems.

The other strength of the Parnas Trace Function Method is the fact that it has wide adoption in industrial case studies of critical systems such as the ones illustrated by

Heitmeyer (2001) and Lawford et al. (2004). TFM representations are simpler to comprehend and are applicable to many more table types than older models discussed in 2.3 but the method is inclusive of previously known and utilized table types. They are also not tied to any specific physical layout, pictures, etc. Also there is broad tool support to assist developers across many generations that allow for developers to deal with traces effectively in the documentation of their software as discussed by Bane et al. (2004) and Eles and Lawford (2011). For the purposes of this research it provides broad coverage to apply the method into the use case of customer due diligence.

4.6 Summary

This chapter surveyed several key technologies in use in financial crime detection systems today. It looked at some of the critical research that has helped develop these technologies and explains why they are important to the financial crime context. Some examples were also provided to show case the complexity of the code necessary to solve this time of problems.

We reason that to potentially use formal methods to check restrictions on specifications of these industry applications we first need to map the underlying technology back to formal methods for appropriateness. We determine that not all areas are appropriate and note that Bayesian (predictive) and neural networks are currently not easily mapped to a formal methods approach and deign them out of scope.

"There's an old story about the person who wished his computer were as easy to use as his telephone. That wish has come true, since I no longer know how to use my telephone."

Bjarne Stroustrup

5

Use Case and Code Review

5.1 Overview

This chapter will examine the financial crime critical system use case introduced previously (Customer Due Diligence). Building on what we know of how financial crime tools operate at a mathematical and functional level we will describe a customer diligence model using the Trace Function Method Parnas and Dragomiroiu (2006).

The emphasis is on the FATF requirements FATF (2012) around customer due intelligence. We will evaluate an application of a code driven mechanism to observe customer attributes and examine the best approaches to describing the program utilizing the trace function method.

5.2 Customer Due Diligence

Customer Due Diligence or *CDD* is a special case of anti-money laundering regulatory compliance that focuses on customer related risk rather than a specific financial transactional risk. The foundation of this domain problem is the exercise of what in business is called *KYC* or "Know Your Customer" type activity. *KYC* is driven by data collection activity both from internal sources and external sources. Through *KYC* a bank is able to make determinations on a how valuable a client's business is to the bank, the specific areas of activity a client engages in and most importantly of all the risk the bank assumes by doing business with this individual or entity.

Customer Due Diligence generally is a modelling-based mechanism that attempts to aggregate information regarding customer data into models and attempt to normalize, cluster and score the information accrued. The basis to all of this is information acquisition at the on-boarding of a newly-acquired client for a given institution into a customer record management system and the ongoing information capture surrounding financial transactions and changes of state in non-financial data such as address, citizenship etc. As an alternative to modelling methods, some vendors try and utilize business rules only to solve for customer due diligence but because of the boolean nature of decision trees are unable to accurately align scoring bands to a degree that meets with regulatory satisfaction.

The end state result of either selected method, modelling or business rules, is an aggregation of scores that represents an overall client risk rating score which in turn presents the overall risk a customer relationship presents for a given institution. Depending on the risk tolerance of that institution they may then decide to 'de-market'/'de-risk' the client, meaning they will cease business with client immediately, close accounts and reimburse balances. When necessary, the bank will also create regulatory filings for regulatory authorities to audit regarding the activity of the client if it is warranted. The results of such filings are also indicators of risk for an ongoing relationship.

5.3 Formal Decomposition

The trace function method employs tabular expressions for the description of component behaviour without explicit reference to internal state (Trancón y Widemann, 2014). Trancón y Widemann (2014) continues by noting this precise form is given by outright course-of-value iteration, for which rough but general implementation guidelines can be given, depending on the pattern of access to the past. The revelation this gives is that through the use of the Trace Function Method we have a path to accurate and systematic representations of software programs.

As discussed in Parnas and Dragomiroiu (2006) that even in the case where a program already exists, asking the users to read the code to find out what it will do in all cases is often inappropriate. There are always cases where people are in a position where the meaning of the code is not easily understandable. Further, often due

to programmatic complexity code can be very difficult and time consuming to grapple with in order to achieve the understanding necessary to modify the code base. This can introduce severe delays in code upkeep and modification by developers tasked with maintaining a software component. This is especially true when we considering the interpretation of complex law into code especially considering the fluid nature of the legislative process and the dynamic nature in which legislators need to respond to newly emerging threats and schemes.

Our goal is examine a software code interpretation of European CDD related legislation and to convert it into a concise, precise and verifiable representation. To achieve this we utilize the Trace Function Method and describe the model in terms of its component variables, interfaces and output function. This chapter is thus focused on studying the use of the Trace Function Method to provide reference documents for professional software developers examining financial crime models.

We will describe a CDD model with the appropriate banding and aggregation rules in the attempt to reach a final score which could be bucketed appropriate for risk management purposes as per the fourth EU Anti Money Laundering Directive EU (2012). This description will use the previously introduced Trace Function Method Parnas and Dragomiroiu (2006) as a basis for formal decomposition of CDD code write in the SAS programming language (Barr and Goodnight, 1976). The aim of this decomposition is to provide developers with a more succinct human readable variant of model documentation which has exhausted all potential outputs¹ and thus becomes viable for formal verification mechanisms.

This model is broken up into several component pieces in order to break down its coverage alignment with the FATF regulations. As such several banding and aggregation mechanisms are used. Banding refers to stratifying the risk score using a logical hierarchy. The typical risk bands utilized in risk rating models are;

High This band is the most severe indication of potential criminal activity or of data elements that could pose either an operational, regulatory or reputational risk to

¹By exhausting all potential outputs we mean that all outputs of the system are known and understood and that the program is incapable of entering into an unknown state.

the institution. Generally, high-risk clients make up only a tiny percentile of the overall client population.

Medium This band is generally utilized when enhanced diligence is required. When flagged this implies that a client presents areas of concern that require further investigation by a compliance specialist. Typically, this occurs generally because of links to sanctions-related risks, high-risk geography linkages or high-risk business activity. The norm here is to establish a continued governance procedure on the client with a regular observation cadence to look for changes in state that would warrant further action. From a client population perspective this is generally a small percentile of the population with a notable bias towards high wealth individuals and business entities.

Low This band is representative of the majority of the client population in normal banking institutions operating in the western world. It indicates little to no correlation with data indicators or behaviours that warrant further investigation and that the client/entity is operating within the normal boundaries of the business relationship.

Aggregation mechanisms are means of combining scores using a logical mechanism to bias in favour or against specific data elements. A classical example of this is the use of *Auto High*¹ or *Panic* elements where in a pool of data elements some are highlighted of being sufficient indicators by themselves of high risks. Recent examples of this would include areas highlighted by FATF FATF (2012) as specifically important such as ties to war torn countries, despots, weapons trading, drug or people trafficking. Otherwise, several elements can be scored in groups and compared algorithmically to determine if the composite score shows a worrisome behaviour.

Often in applications of this nature the use of ceiling and floor functional scores is common in order to add different weights to areas that a bank or country may find of particular importance. Aggregation is used to to combine several different scoring

¹The use of an auto-high indicator is common throughout customer due diligence models as once a key field has been positively validated it may be sufficient of itself to flag a client as a high risk. Examples would include links to terrorist, people trafficking or drug organizations

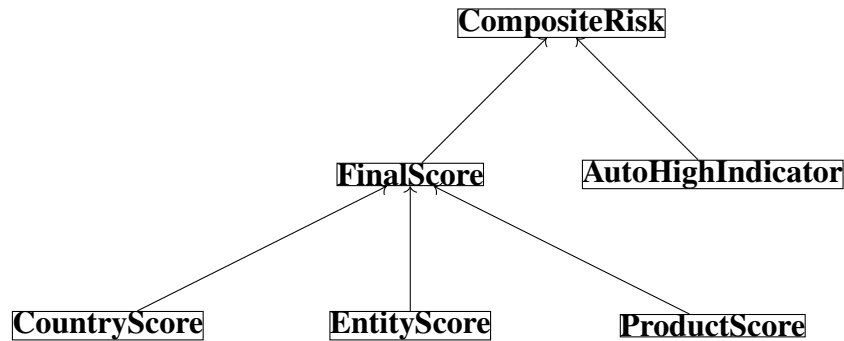


Figure 5.1: Composite Risk Score

elements together to amass an overarching score and can consume either several sub functions or various data elements.

It is also interesting to note that in order to establish the correct band all of the above mechanisms provide scores that correlate back to a band range e.g Low = 0 - 500, Medium = 501 - 750 and High \geq 750. The code and examples provided in the following decomposition are scoped to be inclusive of all relevant elements associated with the 4th EU Anti-Money Laundering direction and do not allow for the proposed changes/amendments undergoing current review in European Parliament EU (2012).

It is often easier to understand aggregations through a graphical reference. The following diagrams introduce the aggregation logic associated with each category pillar and show how each function rolls up to create an aggregate score.

Figure 5.1 elaborates on how the composite risk score is evaluated based on a series of scores coming from each category within the model. These scores propagate up from each child category thereby allowing a bank the flexibility to add new categories as regulation changes. The concept of the *AutoHighIndicator* is to allow the propagation of major risk indicator being positively matched with complete certainty from a specific rule score function to highlight the client in question for immediate review by the banks compliance department.

Figure 5.2 exposes the logic of how the category scores are created. We simplify this representation by examining the rule score functions collectively ($RuleFunc_n$) as the

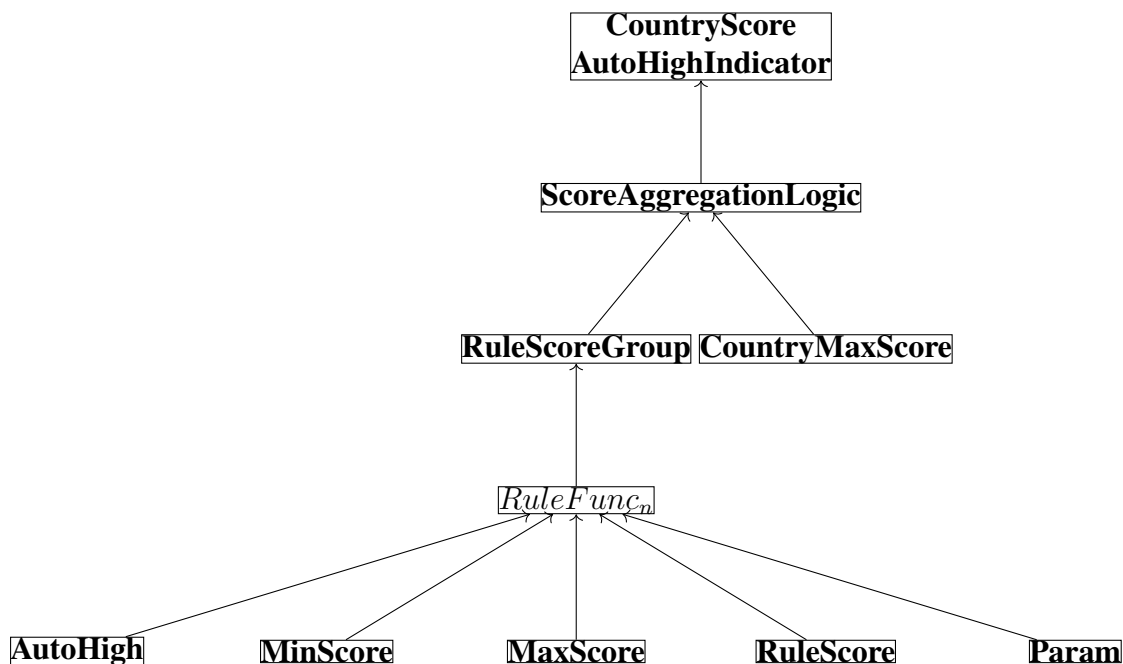


Figure 5.2: Country Score Category

logic can be universally applied to each rule score function. As discussed each function examines a unique compliance related parameter originating from the interpretation of governing legislation EU (2012).

Figure 5.3 and Figure 5.4 show the same logic being applied to the other categories pertinent to the Customer Due Diligence model being examined. This type of rollup allows for ease of modification of the model in the event of a new category pillar being introduced via legislation. Further extension is also available horizontally should new properties regarding a clients activity become required for data capture.

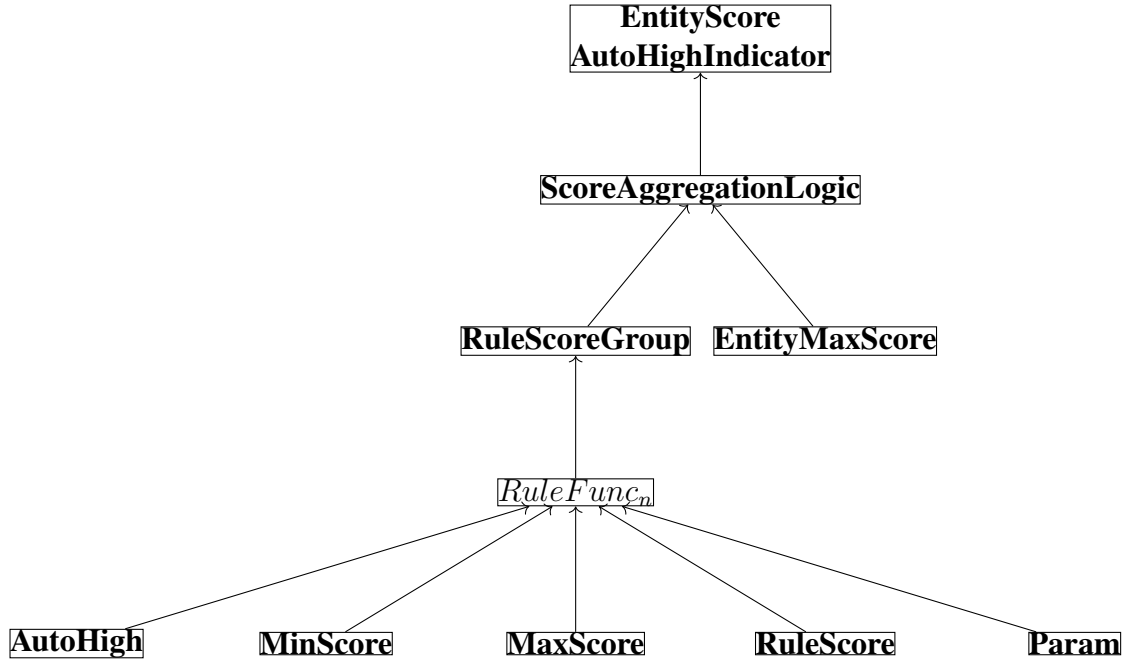


Figure 5.3: Entity Score Category

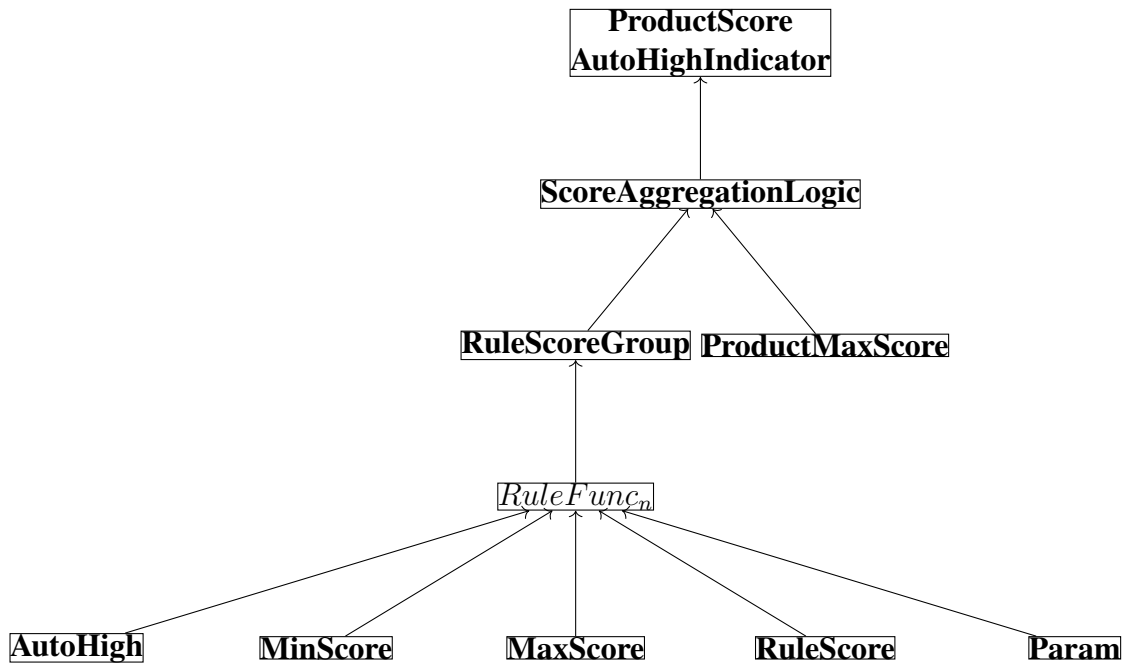


Figure 5.4: Product Score Category

Further, our example code takes a modelling approach with some business rules applied to the aggregation mechanisms. This allows for more flexibility in the model and clarity for the formal description. In the next sections we will attempt to provide the necessary data elements with some description and then conduct a full Trace Function Method description of the elements we now introduce. Coding elements for the associated aggregation rules are available in the first appendix to this work.

The example code elements found in the appendix as well as the tables and rules we explore below are based on an industry example provided by SAS with the approval of their product management leadership. The basis of their existing documentation can be found here; Ferro et al. (2015). Throughout the remainder of this chapter we utilize the IDs and structure that SAS use industrially for supporting Customer Due Diligence in the financial services sector.

5.3.1 CDD Model Overview

As discussed a CDD model is broken into a series of different elements: aggregation rules, data elements and scores. The end result of a CDD model is a score that can be utilized in a customer record management system. The industry norm is to utilize an enterprise case management system that will track changes in customer scores and highlight alarming changes in state. This score can further be applied to banding logic to arrive at the appropriate risk band for the client or entity. In industry these scores are maintained across the entire client population and are continuously updated to record changes in state.

Each area uses different variables with associated variation in types in order to capture all the salient information that the EU requires EU (2012). The general execution work-flow is that data is collected via a relationship manager. The data is digitized at this point to capture personal identifying information like tax numbers, passport number, data of birth etc. This information is centralized to an enterprise data store.

An operational scoring facility consumes this information, assuming that it is of an adequate level of data quality, and scores it against a model deployed in its execution

harness. The cadence of execution for CDD models varies but generally scores are generated via a batch process every twenty four hours. Intra-day execution occurs in small batches to facilitate customer on-boarding events for a bank.

5.3.2 Top Level Aggregation Rules

The CDD model is broken into a series of separate elements in order to facilitate data mapping at a financial institution as every operational data store will have a different data model in place. Top level aggregation rules allow the detection system to prioritize/de-prioritize key elements based on the banks risk appetite. Risk appetite is determined by senior banking officials which combine regulatory factors with profitability motive.

The aggregation rules below divide up the compliance data elements into groupings that can be executed according to the operational profile that the bank has in place. The grouping discussed here cover three main tranches of compliance groups, namely: country, entity and product. Each corresponds to the major compliance risk focus areas where indicators of money-laundering or criminal activity can highlight a client as complicit in some form.

Each aggregate rule group is constructed of three core types. Namely:

Compliance Data Elements These are code elements that specifically grab and parse for hits against indicators that are specifically relevant to AML compliance and client risk rating. These in and of themselves can be grouped also into areas of interest or can be managed individually.

Attribute Maximums This is a ceiling function to set a maximum score value for one of the aggregate groups. These are useful in cases where one pillar outweighs the importance of another and generally occurs where a bank has a higher or lower risk associated with the pillar in question.

Pillar Aggregation This group type does the work of creating the composite score for the group. It does so by analysing the scores generated by each compliance element, combining them and then conducting a comparison to the MAX attribute.

The logic of injecting ceiling and floor values at each level of aggregation allows the risk rating model to offer a lot of flexibility to analysts in mapping the model to the banks offerings and risk appetites.

Country Rule Element Aggregation This group, Table 5.1, ties together areas associated with sanctioned countries and highlights where a client has a tie through business or personal links with a sanctioned country.

Table 5.1: Country Rule Element Aggregation

Rule ID	Rule Name	Order	Rule Group ID	Rule Group Name
CDDR_C101	Registration	1	CDDRG_C010	Country Attribute
CDDR_C102	Primary Business Address	2		
CDDR_C103	Beneficial Owner Residence	3		
CDDR_C104	Director Residence	4		
CDDR_C201	Residence	5		
CDDR_C202	Citizenship	6		
CDDR_C910	Country Max Attribute	1	CDDRG_C020	Country Attribute Max
CDDR_C920	Country Aggregation	1	CDDRG_C030	Country Category Score

Entity Rule Element Aggregation This aggregate group, Table 5.2, examines political exposures for individuals and corporations. Further, it looks at the source of wealth for the associated entity along with ties to shell-organizations, negative news elements. Often terrorist or criminal organizations mask illicit activity through the utilization of charities or companies that have the ability to negate the potential suspicion that one would attribute to certain transactions.

Product Rule Element Aggregation This aggregation facility, Table 5.3, combines several product score areas for the purpose of establishing an aggregate score based on product risk. Product types inherently can be more or less risky. For example, the risk associated with a high value wealth management offering is significantly higher than a credit card with only a \$3000 limit.

Final Score Rule Element Aggregation Each of the previous three aggregation groups presents a score. This group combines those scores and applies some logic around importance, some functional biases and ordering elements. The final score is

Table 5.2: Entity Rule Element Aggregation

Rule ID	Rule Name	Order	Rule Group ID	Rule Group Name
CDDR_E101	Industry	1	CDDRG_E010	Entity Attribute
CDDR_E102	Legal Entity Status	2		
CDDR_E103	Non-Profit Organization	3		
CDDR_E104	MSB	4		
CDDR_E105	Engaged In Internet Gambling	5		
CDDR_E106	Trust Account	6		
CDDR_E107	Foreign Consulate Embassy	7		
CDDR_E108	Issue Bearer Shares	8		
CDDR_E109	Politically Exposed Person (NP)	9		
CDDR_E110	Negative Media News Search (NP)	10		
CDDR_E201	Occupation	11		
CDDR_E202	Politically Exposed Person	12		
CDDR_E203	Negative Media News Search	13		
CDDR_E204	SAR Count	14		
CDDR_E910	Entity Max Attribute	1	CDDRG_E020	Entity Attribute Max

Table 5.3: Product Rule Element Aggregation

Rule ID	Rule Name	Order	Rule Group ID	Rule Group Name
CDDR_E920	Entity Attribute Aggregation	1	CDDRG_E030	Entity Category Score
CDDR_P001	Product	1	CDDRG_P010	Product Attribute
CDDR_P910	Product Max Attribute	1	CDDRG_P020	Product Attribute Max
CDDR_P920	Product Category Aggregation	1	CDDRG_P030	Product Category Score

what would be presented to an operational element where investigators would determine your aggregate risk to the bank. Within that operational platform they would be able to see what elements from the rule groups generated the high scores that pushed a client towards a high risk band.

The final table, Table 5.4, presents the last set of rules associated with the functional elements discussed above. CDDR_X010 creates the aggregate score the previous three groups. CDDR_X015 handles if the bank is dealing with a new or existing customer. CDDR_X020 assigns the customer score to the right risk band. CDDR_X030 is used to determine if this score should be suppressed based on the banks internal policies on risk management. CDDR_X040 handles if the scored customer is going to be sent for further investigation in the form of a case.

Table 5.4: Final Score Rule Element Aggregation

Rule ID	Rule Name	Order	Rule Group ID	Rule Group Name
CDDR_X010	Category Aggregation Rule	1	CDDRG_X010	Category Aggregation Rules
CDDR_X015	New Customer Rule	1	CDDRG_X015	Rule Group Name Strategy Rules
CDDR_X020	Banding Rule	1	CDDRG_X020	Banding/Suppression/Investigation Rules
CDDR_X030	Suppression Rule	2		
CDDR_X040	Investigation Rule	3		

5.3.3 Compliance Element Rule Groups

What follows is an analysis of the different parameters and functions necessary to create the aggregate perspective needed for each pillar. There are groups that directly tie to country, entity and product pillars and beneath each group are associated rule functions. Rather than repeat the code we have eliminated as much redundancy as possible to try and mitigate against mundane elements. All the excluded elements can be found in the appendices of this thesis. The code is included as it was provided by SAS for the purposes of our evaluation. They purport to be transparent and provide the code elements of the model to the client base. This simplifies the documentation for them as all actions can be clearly reviewed assuming that the reader has a knowledge of SAS.

The only variance is the attribute that is under examination and this is extrapolated from EU (2012) and as stated can be reviewed either through reading the Table 5.4.3 of the TFM section or through review of the first Appendix.

5.3.4 Rule Group CDDRG_C010

This rule group examines accounts/clients at a country level against data sets that would contain risk indicators for the bank. These risk indicators would show high risk geographies and potential politically exposed persons.

5.3.4.1 CDDR_C101

This element looks at the registration information for an organization and tries to make a determination if it has links to a high risk geography. It takes the inputs listed in Table 5.5 to manage internal logic. The development team attempt to standardize inputs

where possible to simplify program execution. The inputs provide pointers for the model to examine certain fields of the customer record, in this case REGISTRATION, and give guidance on plugging in values to bias the score created.

Table 5.5: C101 Parameters

Name	Type	Description	Value
ATTR_NAME	Character List	Attribute Name	"REGISTRATION"
MAX_SCORE	Numeric Constant	Maximum Score	200
MIN_SCORE	Numeric Constant	Minimum Score	0
WEIGHT	Numeric Constant	Weight	1

SAS The code used in the following example is constructed in the language of SAS (Barr and Goodnight, 1976). The SAS language provides mechanisms for designing complex models and working with data. Further, SAS is an accomplished vendor in the financial crime detection solution world and has its solutions deployed at many financial institutions internationally.

This code is what is used by the model execution engine, in this case SAS, in order to arrive at a score for a certain category. It utilizes the variables discussed above and is applied to an incoming data set of banking information regarding a client base. The code associated with the other rules discussed in this chapter is available in the first appendix of this work.

```
%cdd_rule_attr_level( attribute_code=&attr_name
, party_type_desc='ORGANIZATION'
, weight= &weight
, max_score=&max_score
, min_score=&min_score
, auto_high_score=
, rule_score_exp=LOOKUP_SCORE
, not_exceed_range_YN=Y
, auto_high_exp=%nrquote(ifc(RULE_SCORE > MAX_SCORE, 'Y', 'N'))
, triggering_value_var=ATTRIBUTE_CHAR_VALUE
, triggering_value_type_code=CHAR
);
```

5.3.5 Rule Group CDDRG_C020

A sanctions rule group is one that examines for breaches of national and international sanctions on people, goods and countries. Not every sanction rule group contributes the same level of risk to the bank. In our example the developer simplified matters by creating one group of rules to map to sanctions regulations in EU (2012). However in a situation where others exist the use of minimums and maximum values would be more of a necessity.

5.3.5.1 CDDR_C910

No parameters are utilized here as this manages the maximum value attributed to the overall group. Once all values from each rule within the group are aggregated the score is checked against the maximum value here and if it is exceeded the model uses the score here otherwise the aggregate number. Here a bank would be able to decide and adjust the overall risk attributed to country-based KYC risks and make a determination if it is more/less important to the operations of the bank. Normally a bank would determine this area to be of higher importance when they have more of a multi-geographic presence or if they operate within a region where there is extreme risk of terrorist group activities.

```
/* Country Max Attribute Rule */  
%cdd_rule_attr_max;
```

5.3.6 Rule Group CDDRG_C030

The *country* category group will always be a dynamic category due to the changing nature of sanctions lists. As new lists and risks are identified the logic controlled how those scores impact country risk scores would be considered in this section.

5.3.6.1 CDDR_C920

No Parameters are necessary for this group as it creates the composite score based off of the guiding logic of the maximum and the overall combinatorial logic of the sub-groups. The *PRIMARY_ENTITY_NUMBER* referred to below is indicative of a unique

identifier that the bank would use to identify all client activity at the bank through any channel. The code behind this group behaves by doing the actual aggregation of all necessary rules for this specific client.

```
/* Weighted Sum of Country Attribute Scores Divided by
   Weighted Sum of Max Country Score */

/* Weight should be set to 1 if the category level score
   should not be weighted. */
/* A different weight can be applied when calculating the
   overall score. */

%cdd_rule_cat_agg( weight=1
, by_list=PRIMARY_ENTITY_NUMBER
, score_type_code="CAT"
, rule_score_exp=%nrquote(ifn(SUM_MAX_SCORE ne 0, SUM_RULE_
  SCORE*WEIGHT / SUM_MAX_SCORE*100, .))
, auto_high_exp=MAX_AUTO_HIGH_IND
);
```

5.3.7 Rule Group CDDRG_E010

This grouping examines entity related risk. As previously discussed this type of risk examines compliance indicators for both personal and legal entities (e.g. corporations) and examines them for areas where illicit activity is known to occur. Money launderers attempt to use case intensive businesses along with mechanisms where money can be issued to non-identifiable owners in order to clean money. Some vehicles of these types of activities include casinos, money service bureaus and shell corporations. At a personal level it also includes relatives and close associates of political entities in troubled governments that can be used as influencers in corrupt practices.

Further, certain types of industrial activity is indicative of potential high risk activity. Normally people would associate this type of activity with weapons trading but even construction activity can mask illicit practices. The rules here follow the same structure

as the preceding category and thus we exclude all other rules with the exception of the first to avoid redundancy.

5.3.7.1 CDDR_E101

This group focuses on industry associated with the specific entity and how it generates wealth. The bank will utilize industry type tables and will associate risk values with each industry based on its overall risk appetites. Examining the input variables, Table 5.6, and focusing on the important attribute under examination we can see that here the clients area of employment is being scored against a list of industry codes.

Table 5.6: E101 Parameters

Name	Type	Description	Value
ATTR_NAME	Character List	Attribute Name	"INDUSTRY"
WEIGHT	Numeric Constant	Weight	1
MIN_SCORE	Numeric Constant	Minimum Score	0
MAX_SCORE	Numeric Constant	Maximum Score	100

```

/* Industry Attribute Rule */
%cdd_rule_attr_level( attribute_code=&attr_name
, party_type_desc=' ORGANIZATION'
, weight=&weight
, max_score=&max_score
, min_score=&min_score
, auto_high_score=
, rule_score_exp=LOOKUP_SCORE
, not_exceed_range_YN=Y
, auto_high_exp=%nrbrquote( ifc( RULE_SCORE > MAX_SCORE, 'Y', 'N'
) )
, triggering_value_var=ATTRIBUTE_CHAR_VALUE
, triggering_value_type_code=CHAR
);

```

The remain rules and rule groups focus on the mechanics of generating a score and operate in the exact same way as the previous section.

5.3.8 Rule Group CDDRG_P010

This rule examines banking products from a risk management perspective. In this case products can be various forms of credit instruments like cheques, card accounts etc. This looks cumulatively at the product utilization of a client/legal entity in order to risk rate their modus operandi in banking. This score is further normalized through segmentation mechanisms where the client is compared to similar individuals/entities in order to establish a behaviour pattern that they risk rate as normal.

5.3.8.1 CDDR_P010

This rule examines all the products that a bank offers and rates it against a risk ranked scale of how the bank assesses them. The bank changes this hot-list often as new products are created/retired and as a result is often in flux. Given the changing nature of banking, moving from a physical interactions to digital, banks are increasingly wary of ensuring more security controls for product offerings in order to meet regulatory needs. This however counteracts customer experience measures and profitability as it inhibits a clients ease of use for new banking products. Table 5.7 is the manifestation of this in the model.

Table 5.7: P010 Parameters

Name	Type	Description	Value
ATTR_NAME	Character Constant	Attribute Name	"PRODUCT"
MAX_SCORE	Numeric Constant	Maximum Score	200
MIN_SCORE	Numeric Constant	Minimum Score	0
WEIGHT	Numeric Constant	Weight	1

```

/* Weight should be set to 1 if the category level score
   should not be weighted. */
/* A different weight can be applied when calculating the
   overall score. */

%cdd_rule_cat_agg( weight=1
,by_list=PRIMARY_ENTITY_NUMBER
,score_type_code="CAT"

```

```

, rule_score_exp=%nrquote(ifn(SUM_MAX_SCORE ne 0, SUM_RULE_
  SCORE*WEIGHT / SUM_MAX_SCORE*100, .))
, auto_high_exp=MAX_AUTO_HIGH_IND
);

```

5.3.9 Rule Group CDDRG_X010

The purpose of this rule group is to start combining all of the three categories into a composite score. Business analysts at a bank would have the opportunity to further tune the bias of different elements at this level. Operationally the bank looks to create risk buckets for all the client population that are tuned and examined on an ongoing basis. This grouping allows them the flexibility to augment the scoring categories with new ones should they deign them necessary.

5.3.9.1 CDDR_X010

The output of this group would be the final risk score associate with the client or legal entity. This score would then pass into more operational rules to assign it to a risk band for investigators to examine. This score is the aggregate value passed from the three major categories as per Table 5.8. The code here functions by first doing some basic setting and getting of values. Next by examining the customer record by doing a lookup on their PRIMARY_ENTITY_NUMBER it starts the scoring process by checking if AUTO_HIGH_IND is operating within expected conditions. It then constructs the score for each of the three categories; COUNTRY, ENTITY and PRODUCT. The RULE_SCORE is and AUTO_HIGH_IND take the appropriate values and output is generated to the appropriate data store.

Table 5.8: X010 Parameters

Name	Type	Description	Value
COUNTRY_WEIGHT	Numeric Constant	Country Weight	0.4
ENTITY_WEIGHT	Numeric Constant	Entity Weight	0.35
PRODUCT_WEIGHT	Numeric Constant	Product Weight	0.25

```
/* Calculate Sum of Weighted Category Scores */
length TEMP_RULE_SCORE 8.;
retain TEMP_RULE_SCORE 8.;
drop TEMP_RULE_SCORE;

length MAX_AUTO_HIGH_IND $ 1;
retain MAX_AUTO_HIGH_IND;
drop MAX_AUTO_HIGH_IND;

if first.PRIMARY_ENTITY_NUMBER then do;
TEMP_RULE_SCORE = 0;
MAX_AUTO_HIGH_IND='';
end;

if AUTO_HIGH_IND > MAX_AUTO_HIGH_IND then MAX_AUTO_HIGH_IND=
    AUTO_HIGH_IND;
if CATEGORY_CODE = "COUNTRY" then TEMP_RULE_SCORE = TEMP_RULE_
    SCORE + RULE_SCORE * &country_weight;
if CATEGORY_CODE = "ENTITY" then TEMP_RULE_SCORE = TEMP_RULE_
    SCORE + RULE_SCORE * &entity_weight;
if CATEGORY_CODE = "PRODUCT" then TEMP_RULE_SCORE = TEMP_RULE_
    SCORE + RULE_SCORE * &product_weight;

if last.PRIMARY_ENTITY_NUMBER then do;
CATEGORY_CODE='';
RULE_SCORE = ROUND(TEMP_RULE_SCORE,1);
AUTO_HIGH_IND=MAX_AUTO_HIGH_IND;
output;
end;
```

5.3.10 Rule Group CDDRG_X015

An important distinction in risk management for banks is the treatment of new clients at on-boarding versus ongoing due diligence of the establish client base. As such the rigour and intelligence gathering of new clients being on-boarded into the

banks core systems is a more onerous activity as products are being established by the relationship manager. Information collection on existing clients at the bank can be made difficult to track as it is at the discretion of the client to provide updates to changes in status but the majority of the client population are transparent in their fiscal dealings.

5.3.10.1 CDDR_X015

This rule group examines the customer age and allows the bank to tweak treatment based on the client establishment date, i.e. the date the client registered their first product with the institution in question. Table 5.9 looks at the weighting that a bank would give to this process given the information, or lack thereof, provided. Further, it specifically measures the age of the relationship with the client in question and applies different treatments accordingly. This is reflected in the code snippet below which looks specifically at the field CUSTOMER_SINCE_DATE which it pulls from the customer record. In the absence of a date in this field the program reacts by setting a date record for today and weighting the customer using the preset weight bias the bank has for new customers.

Table 5.9: X015 Parameters

Name	Type	Description	Value
new_customer_age	Numeric Constant	Age in years customer NOTEQUALTO new	1.01
new_customer_weight	Numeric Constant	weight to use for a new customer	1.2
missing_age_weight	Numeric Constant	weight for customers with unknown on-boarding date	1.2

```

/* Apply weight based on customer age */
if RULE_SCORE ne . then do;
if CUSTOMER_SINCE_DATE eq .
then do;
/*use missing date weight*/
RULE_SCORE = ROUND(RULE_SCORE * &missing_age_weight,1);
end;
else do;
if yrdif(datepart(CUSTOMER_SINCE_DATE), today(), 'AGE') <= &
    new_customer_age
then do;

```

```
RULE_SCORE = ROUND(RULE_SCORE * &new_customer_weight,1);  
end;  
end;  
end;
```

5.3.11 Rule Group CDDRG_X020

This rule group manages finalized risk score mechanics for operational investigation facilities at the bank. Banks hire compliance specialists tasked with ongoing monitoring of the client base. They analytically determine what the banks risk tolerance should be. For ease of use once these bands are analytically determined based off the populations initial scores they are provide as ranges/bands that are highlighted via colour schemes in enterprise case management systems. An enterprise case management system is a workflow system to track changes in scores with the client population.

5.3.11.1 CDDR_X020

This rule group allows the bank to set minimum and maximum ranges for risk bands with the client base. They can fine tune these with ease at any point and even create new bands for consumption at a workflow level if needed. Table 5.10 sets the banding score limits for the model to determine the appropriate boundaries between the different risk buckets. The code snippet provided by SAS shows more detail of how this works as it lists out the classification methodology. What occurs is the checking of RULE_SCORE against the different risk bucket values; l_max, and m_max. Should the value exceed the ceiling value of m_max it is clearly in the HIGH risk bucket. Logic is also provided here that if AUTO_HIGH_IND has a character value of yes then the risk is immediately set to high. Some other logic included here is to flag what the last review score was and record the difference for analysis. Finally the score is persisted.

Table 5.10: X020 Parameters

Name	Type	Description	Value
L_MAX	Numeric Constant	Low Risk Max Score	50
M_MAX	Numeric Constant	Medium Risk Max Score	85
SCORE_DIFF	Numeric Constant	Score Difference	10

```

/* Assign Score Classification */
if RULE_SCORE <= &l_max then SCORE_CLASSIFICATION_CODE = 'L';
else if RULE_SCORE <= &m_max then SCORE_CLASSIFICATION_CODE =
  'M';
else SCORE_CLASSIFICATION_CODE = 'H';

/* Auto High flag overrides score classification */
if AUTO_HIGH_IND eq 'Y' then SCORE_CLASSIFICATION_CODE = 'H';

/* Use the override rating if the new score is close to the
   last review score */
if CDD_REVIEW_FLG ne '1' and
SCORE_CLASSIFICATION_CODE='H' and
OVERRIDE_SCORE_CLASS_CODE ne SCORE_CLASSIFICATION_CODE and
OVERRIDE_SCORE_CLASS_CODE ne '' and
( RULE_SCORE <= LAST_REVIEW_SCORE or
(RULE_SCORE - LAST_REVIEW_SCORE) <= &score_diff
)
then do;
SCORE_CLASSIFICATION_CODE = OVERRIDE_SCORE_CLASS_CODE;
end;

```

5.3.11.2 CDDR_X030

Operationally decisions need to be made around client retention versus de-risking/de-marketing them should they hit sufficient levels of risk scores. It is important to ensure that this is flagged within the system so that in the case the client tries to re-establish a business relationship the past scores are persisted. Further, in the case of a retention de-

cision certain aspects of the ongoing scoring mechanisms can optionally be suppressed in order to allow the bank a means of 'ignoring' the client until a major change occurs.

The value SCORE_DIFF, Table 5.11 highlighting the variance of a change in a customers risk categorization is the basis of activity here. This code is specifically used within the workflow investigation system for the CDD case review process. The outputs here are set directly in the workflow system to decide what action should occur. While workflow is an interesting problem to solve we are de-scoping this aspect from the TFM as it would require a whole program of other work to capture the data model used in full.

Table 5.11: X030Parameters

Name	Type	Description	Value
SCORE_DIFF	Numeric Constant	Score Difference	10

```

/* Suppression Rule */

if FINAL_DISPOSITION_CODE='RETAIN' and
CDD_REVIEW_FLG ne '1' and
SCORE_CLASSIFICATION_CODE='H' and
( RULE_SCORE <= LAST_REVIEW_SCORE or
(RULE_SCORE - LAST_REVIEW_SCORE) <= &score_diff
) then do;
CDD_SUPPRESSION_FLG='1';
end;
else do;
CDD_SUPPRESSION_FLG='0';
end;

```

5.4 TFM

As discussed earlier and as defined per Bane (2008) the trace function method offers application developers a means of describing programs mathematically. There are two

major advantages to the method: firstly, it allows a developer to minimize the documentation elements associated with a program. As we've seen from our industrial example documentation to describe the complexities of a program can be time intensive activity. Secondly, it allows developers to verify formally that the program acts as specified and does not result in unexpected outcomes. The TFM is built on early tabular notation formats as per Janicki and Khedri (2001) but has evolved from the basic tabular notations Parnas used in Parnas et al. (1987) to have more of a trace based focus. The notation now focused more on traces, events, programs, messages and set elements and is more reflective of how technology has changed in the intervening forty years since formal methods started to be used.

To show the power of the trace function method we utilize an industrial example, the customer due diligence model previously introduced. In doing so our goal is to attempt to minimize the associated program specification whilst maintaining the comfort of being able to validate all outputs of the system. This ensures that any model meets with the formal regulatory references provided by regulations like the 4th EU anti money laundering directive EU (2012) and the recommendations proposed by FATF FATF (2012). What regulators would expect to see here is a readable format of what fields are being examined in customer records, how scores are generated and to garner some idea of how the program logic would work. Someone in more of an IT focused role would expect to learn more about what would drive an error state and how it could be avoided. Both use cases are supported through the TFM's layout.

5.4.1 Using the TFM

It should be noted that earlier in this work we provided a basic example to explain some concepts of how tabular notation and the TFM operates in 2.1. This was a basic example that purely sought to describe a normal function table and introduce the topic rather than the complete details necessary to create an accurate TFM component description document. Other examples that align more to how our industrial use case is described include; Quinn (2007), Parnas and Dragomiroiu (2006) and Parnas (2006). The following example is, as previously noted, a real example from industry and has

been successfully deployed as an active model is several banking institutions around the globe.

The TFM will be used in this context to explain the relationship between several elements of the model and their associated variables. It is important that we note the context of that relationship graphically in order to highlight how the TFM simplifies these links. The purpose of the program is to create a composite risk value of a given entity/client in order to determine if they pose a compliance risk to the bank.

As such the program is designed to use several major categories that aggregate up using predetermined logic to bias the scoring algorithms. This priority bias would normally be provided by the bank where the financial crime detection system is deployed. For the purposes of this example we assume that each pillar is of like importance. Built into the logic of the model is the ability to bias at all levels allowing them to create floor and ceiling values for both the category and the rule scoring functions that are members of each category.

Parnas and Dragomiroiu (2006) gives us a definition of what a TFM component interface document comprises of:

- a complete description of the component's inputs (their type).
- a complete description of the component's outputs (their type).
- a description of a set of relations, each one describing the relation of the value of an output to the history of the values of the inputs. The range of each relation is the set of possible values for the associated output variable. The domain of each relation is a subset of the set of all possible histories for that component.

Elaborating beyond this it is noted that if the behaviour being documented is deterministic, the relations will be functions. Further, histories should include all past behaviour including the actual output values as well as the actual input values; in some methods, non-determinism causes difficulty because one can only refer to input values (Parnas and Dragomiroiu, 2006).

5.4.2 Component Programs and Inputs

The first point of reference is the definition of all program¹ components or modules (Parnas, 1972a) defined within the model. These programs in the case of our CDD model breakout according to each category examined within CDD.

Beginning with the most important element of the CDD software model's function the first program examined is Table 5.12. This program is responsible for the final aggregation of scores from all CDD category pillars. As we are looking at scores the associated type of each element of the program is an integer type. The ranges are generally determined by a given financial institution but generally will follow a pattern like zero to one hundred or zero to one thousand.

The associated range values can then be bucketed by an institution to highlight risk level and generally the resulting curve should show the majority of the client population in low risk categories with much smaller percentages being in medium to high risk categories.

Table 5.12: PGM(FinalScore)

Name	Type	Range
COUNTRYScore	< int >	
ENTITYScore	< int >	
PRODUCTScore	< int >	

The program we evaluate next is the one responsible for country score element aggregations. As noted from earlier in the chapter, country score is a category of the CDD model that examines data elements within the client populations information pertinent to addresses, places of birth, residence and citizenship etc. These are examined thoroughly against lists of countries that have a diplomatic sanction against them to highlight that a client is a potential risk for the bank.

¹As the concept of the module evolved around specific work tasks the terminology evolved to be inclusive of language like software components or programs.

The component program, Table 5.13 encapsulates some extra logic around variable weighting for the convenience of the banks compliance department. Not all risk variables are of equal importance. Thus a bank needs to be able to adjust the model to take into account its subjective view of the risks to its business and align the weights accordingly.

Further, there is the ability to allow both a floor and ceiling values to certain variables once again based off the banks overall business model and risk tolerance. These last elements have a clear impact on the range of values associated with the variables that vary bank to bank.

Table 5.13: PGM(CountryScore)

Name	Type	Range
ATTR_NAME	< enum >	
MAX_SCORE	< int >	
MIN_SCORE	< int >	
WEIGHT	< int >	

The component program, Table 5.14 follows the same structure as its predecessor, Table 5.13, but differs in domain context. As previously discussed this program component area focuses on private individuals and corporate entities. It examines several areas of their business dealings and associates a risk value with them. The same mechanisms to set weights, ceiling and floor values also exist here to ensure flexibility in the deployed model.

Table 5.14: PGM(EntityScore)

Name	Type	Range
ATTR_NAME	< enum >	
MAX_SCORE	< int >	
MIN_SCORE	< int >	
WEIGHT	< int >	

The next program in the series, Table 5.15, also follows the structure of the last two programs. The domain in this case examines the product mix that a client may be taking advantage of from the banking institution i.e. lines of credit, corporate loans, commercial payments, correspondent banking, wealth management. These products have different risk ratings and this program allows the banks analysts to adjust them accordingly. Mechanically, this program follows the same logical functions as the previous two programs.

Table 5.15: PGM(ProductScore)

Name	Type	Range
ATTR_NAME	< enum >	
MAX_SCORE	< int >	
MIN_SCORE	< int >	
WEIGHT	< int >	

The final key program in the CDD model is presented in Table 5.16. This program is a mechanical means of showing an exact match with a high risk concern. It allows for any client to be flagged as mapping completely to a country, entity or product category risk. Should this occur this function allows for an over-ride that is surfaced via a case management work-flow that an analyst would be able to investigate.

Structurally we are looking at a flag variable for the associated client and a message lock from the originating data point that highlighted the auto high.

Table 5.16: PGM(AutoHigh)

Name	Type	Range
Message Name	Flag	
AUTO_HIGH_IND	< Char >	

The purpose of this section of the TFM is to provide a very concise and clear overview of what each component programs variables are, their types and associated ranges. Through this specification we also can ascertain some of the logic that is necessary but this is really only apparent because of the labelling we have used in this

use case. Plain simple language is a benefit in any specification style and as such this should not act as a biasing element in a decision to use the TFM.

5.4.3 Set Elements

In this section we will examine all program data elements examined by the CDD model and used within its code base for the purposes of validating compliance. These set elements align directly to the legislation requirements derived from EU (2012). As observed from the previous section these elements are consumed and compared against the banks client data set and the associated attributes contained therein. The other consumed data source is global watch lists that contain data elements that the scoring harness uses during processing.

Beginning with Table 5.17 this set of elements consumed is made up of some domain assets and some logical assets that are used in the associated program, Table 5.13, previously discussed. The focus here is sanctioned country data attributes.

Table 5.17: Elements of Set CountryScore

Element Name
ORGANIZATION
REGISTRATION
STREET
MAILING
DIRECTOR_RESIDENCE
OWNER_PCT
PRIMARY_ENTITY_NUMBER
MAX_AUTO_HIGH_IND
CAT
LOOKUP_SCORE

The next set of elements, Table 5.18, follows the exact same pattern as Table 5.17 in that it is a set of domain and logical assets used by the program Table 5.14. The focus area is risk associated with entities such as individuals and corporations. The set elements here reflect that focus and highlight concerns that regulators have with this aspect of due diligence.

Table 5.18: Elements of Set EntityScore

Element Name
INDUSTRY
ORGANIZATION
LEGAL_ENTITY_TYPE
NON_PROFIT
MONEY_ORDERS
MSB
PREPAID_
TRAVELERS_CHEQUES
CURRENCY_EXCHANGE
INTERNET_GAMBLING
TRUST_ACCOUNT
FOREIGN_CONSULATE_EMBASSY
ISSUES_BEARER_SHARES
PEP
NEGATIVE_NEWS
OCCUPATION
INDIVIDUAL
SAR_COUNT
MAX_AUTO_HIGH_IND
CAT
LOOKUP_SCORE

The final set of elements, Table 5.19, provides the set of data assets necessary for the final program component discussed earlier, Table 5.15. Product elements are a much smaller asset tailored to each bank and their marketplace and clientele.

Table 5.19: Elements of Set ProductScore

Element Name
PRODUCT
MAX_AUTO_HIGH_IND
CAT
LOOKUP_SCORE

Through the provision of these set elements the TFM provides developers with an easily maintainable list of all data of interest to the model. In the case of our CDD use

case this is simplified from being one big long list by the fact that the model is broken into specific pillars. This allows for easier consumption based off of the domain aspect of CDD that each category is solving for. Once again though this provision is more of a best practice activity rather than a specific designation within the TFM itself.

That being said, the availability of these lists allows developers to easily check and validate that they are accounting for key elements of legislation being examined. The lift this approach provides is that augmentation of the specification is greatly simplified meaning that as new legislation occurs an augmentation strategy for the code base can easily be aligned to the specification change without negatively impacting the execution of the model.

5.4.4 Auxiliary Functions

Auxiliary functions in this case are representative of the mechanisms used to create the composite scores within the CDD model. Further, they also are used for the logic to control minimum and maximum values for scoring elements for each category group.

We originally described Tables 5.12, 5.13, 5.14, 5.15 and 5.16 as the high level of program components that make up the model we are describing. We next examine the auxiliary functions that are the actual worker elements for these programs. These represent the underlying functions that carry out this work within the model.

Table 5.20 provides a list of the auxiliary functions that provide the logical work discussed in the previous section. These auxiliary functions are responsible for acting as setters and getters for the different scoring elements that these programs require. Specifically this listing address country related category functions.

Table 5.20: CountryScore Auxiliary Functions

i	CountryScore(i)
1	CDDRG_(C010)
2	CDDRG_(C020)
3	CDDRG_(C030)

Table 5.21 provides the same functional listing as Table 5.20 provides but does so for the entity category. To elaborate this allows for easily bread-crumbing for a developer to track data elements and functions through the TFM. They can quickly ascertain in this case where to look in the specification for functions pursuant to entity related scoring by using this simplified index.

Table 5.21: EntityScore Auxiliary Functions

i	EntityScore(i)
1	CDDRG_(E010)
2	CDDRG_(E020)
3	CDDRG_(E030)

Table 5.22 follows the same pattern as both Table 5.20 and Table 5.21 in providing a listing that is centric to auxiliary functions that are necessary for the product category of the CDD model to function as designed.

Table 5.22: ProductScore Auxiliary Functions

i	ProductScore(i)
1	CDDRG_(P010)
2	CDDRG_(P020)
3	CDDRG_(P030)

With this listing of auxiliary functions the TFM is giving developers access to a functional index of what functions are available for use within the program components of the CDD model. Once more, it helps that this use case has already separated out the categories as it simplifies the auxiliary function listing. In this case the function names are not very transparent and provide no semantics on what they do at this level of obfuscation. That detail will come at later levels of decomposition.

5.4.5 Output Messages

This section of the TFM specification contains the absolute outputs of the software components being specified/described. In the case of the CDD model we are describing

there are two to note, namely, the final score and an auto high indicator. The latter is a mechanism used in CDD to alert investigators that there was a high correlation match with against the customer data and that the client needs to be treated as high risk.

Table 5.23 presents the messages that are being written to each customer profile that the model is processing. The intent of the model is to append each customer record with the appropriate value with the end state being an accurate map of risk scores across the entire customer population. These outputs would be consumed by a workflow case management system to track changes in customer scores and to act upon these changes in state.

Table 5.23: Set of Output Messages

Element Name
FINAL_SCORE
AUTO_HIGH_INDICATOR

The focus of the TFM here is to allow a developer or any reader of the software specification to be able to easily check the intended outputs of the software. Once again this is presented in a very clear and concise fashion. The variables in question though are helped through the naming convention and standards used in this domain model.

5.4.6 Output Function

Under the context of the TFM specification, the purpose of the output function is to create a summarized variant of the program using a trace. These traces provide a sequence of expected actions for the associated functions. It further decomposes the earlier indices to this final level where all the mechanics are exposed. The representation should show the path for success and also paths to error states.

This representation can be utilized in formal verification processes for formal verification purposes. Earlier work in tabular expressions created tools backed by mechanized mathematical tools for this purpose Jing (2000). The majority of these tools utilized theorem proving systems like PVS Shankar et al. (1993).

With a verified output function developers are able to check for completeness and disjointness on the output function and thereby provide confidence around the potential states that the program can enter into and outputs it can create. For critical systems this is highly advantageous and helps against the manipulation of the program into unforeseen applications.

As our example is effectively a series of aggregate model elements we can break the subsequent categories up into their effective program elements. Each rule score is an isolated element of common form focused on one data element within the category as such we can describe this code once and sum where appropriate in the aggregation group. With each category aggregated we can conduct a final logical aggregation. The final point of order is the management of the `AUTO_HIGH_INDICATOR` which is a fundamental element which cascades from all rule score functions. The logic is thus, if any rule score function exhibits an immediate match without any potential data misrepresentation then the variable should flag as high and tag the entity accordingly.

5.4.6.1 Country Category Output Functions

These functions are specific to the generation of values associated with the CDD model for the country category of risks that a banks client base will be evaluated against. We will describe each function using the TFM.

Table 5.24 presents us with a trace that describes how the program arrives at the outcomes for `RULE_SCORE` and `AUTO_HIGH` for the CDD pillar examining country risk. The TFM creates an omnibus view of what the program is trying to achieve along with all potential error states. Thus, below we are presented with traces(T) that provide the path to generating the values for each. Where $last(T)$ is an auxiliary function to find the corresponding output message for the trace T in valid situations that do not result in an error state.

Table 5.24: Output Function Country Group Rule Score Function

	T	O(T)
last(T) INSET RULE_SCORE^	last(T) = (WEIGHT*(RULE_SCORE_EXP+LOOKUP_SCORE)) last(T) NEQ ((WEIGHT*(RULE_SCORE_EXP+LOOKUP_SCORE)))^	RULE_SCORE ERROR
last(T) INSET AUTO_HIGH^	last(T) = (RULE_SCORE > MAX_SCORE) last(T) NEQ (AUTO_HIGH)^	AUTO_HIGH_INDICATOR ERROR

Table 5.25 is a trace representation of the function responsible for setting a max value for CDD country pillar attributes, the logical ceiling value, to allow for tailoring the model at a specific banking institution.

Table 5.25: Country Max Attribute Value

T		O(T)
last(T) INSET COUNTRYATTRIBUTEMAX^	last(T) = (CDD_RULE_ATTR_MAX) last(T) NEQ ((CDD_RULE_ATTR_MAX))^	COUNTRYATTRIBUTEMAX ERROR

Table 5.26 is the trace that gives us the view into how the final aggregate score, *COUNTRYScore* is generated. It shows the summation of all generated *RULE_SCORE* values while accounting for the logical bounding mechanics that are unique to a given bank.

To summarize what we have represented above using the TFM; we have documented using the TFM specification the traces of functions necessary to create an aggregate score for the attribute rules defined earlier specific to the country related risks defined in CDD legislation. These traces show how bounding logic, aggregation and error mitigation come together in the TFM to show how a score is generated and managed. Further, it provides a developer with a reference document to generate a program to solve for this agnostic of development language.

5.4.6.2 Entity Category Output Functions

These next set of functions are specific to the generation of values associated with the CDD model for the, previously described, entity category of risks that a banks client base will be evaluated against. We will describe each function using the TFM.

Table 5.27 describes using the TFM the process of generating a rule specific score for the variable *RULE_SCORE* and whether or not an *AUTO_HIGH* has been triggered.

Table 5.28 behaves in the same way as Table 5.25 and acts as a logical boundary to control the values so that the bank can adapt to model to its own risk appetite.

Table 5.26: Country Category Aggregate Score Logic

	T	O(T)
last(T) INSET COUNTRYScore	$\text{last}(T) = (\text{SUM_MAX_SCORE} \neq 0, \sum_{i=1}^n \text{RULE_SCORE}_i * \text{WEIGHT} / \text{SUM_MAX_SCORE} * 100)$	COUNTRYScore
last(T) INSET AUTO_HIGH	$\text{last}(T) \neq (\text{SUM_MAX_SCORE} \neq 0, \sum_{i=1}^n \text{RULE_SCORE}_i * \text{WEIGHT} / \text{SUM_MAX_SCORE} * 100)^{\wedge}$ $\text{last}(T) = (\text{AUTO_HIGH_INDICATOR})$ $\text{last}(T) \neq (\text{AUTO_HIGH_INDICATOR})^{\wedge}$	ERROR AUTO_HIGH_INDICATOR ERROR

The final function of note within the entity category is Table 5.29 which behaves in a similar fashion to Table 5.26. The TFM representations shows how the final aggregation of the entity scores occurs whilst accounting for logical conditions applied to them. It also shows potential error states.

5.4.6.3 Product Category Output Functions

The final category in the CDD model we are describing is the product category. As before we will describe each functional element using the TFM and consider the traces for each.

Table 5.30 behaves as the previous two category rule score aggregations (Table 5.24 and Table 5.27). Thus the TFM representation is similar. There would be an opportunity to collapse elements of this specification but due to the design decision to bundle by category in our industrial example we have to ensure an accurate mapping.

Table 5.31 presents the TFM representation of the attribute value functions that bound the max value that can be attributed in this case to a specific product attribute e.g. The risk of the cheque products is low in comparison card products so attribute maximums are set accordingly.

Table 5.32 is the TFM specification of the final aggregation scoring function associated with aggregating all the product rule scores and bounding conditions. Here, as per the previous categories, the TFM shows the summation of all product rules to arrive at that final score along with articulating the error state. Further, it accounts for the flagging of the `AUTO_HIGH_INDICATOR` on the product category for use within an enterprise case management system at a specific bank.

This final category followed the pattern of the previous ones ultimately showing the possibility of collapsing the specification down for the purposes of the TFM representations. Arguably though because we are working with a pre-drafted industrial example we have followed the design decisions made by the developer rather than have an expectation that a code variant of the domain problem be derived from the specification provided here.

5.4.6.4 Final Program Output Function

The purpose of a CDD model is to score. This final function is the end point of all previous aggregation mechanisms and takes into account all category scores. It also provides a final tally of for the presence of an AUTO_HIGH_INDICATOR flag such that the model will tag all clients with a score and add the flag as necessary.

Table 5.33 provides us with that final perspective of the model in TFM terms. It shows the specification aggregating the categorical variable elements as designated and shows the propagation of the flag variable from categories to this final evaluation of the client.

Table 5.27: Output Function Entity Group Rule Score Function Template

T		O(T)
last(T) INSET RULE_SCORE [^]	last(T) = (WEIGHT*(RULE_SCORE_EXP+LOOKUP_SCORE))	RULE_SCORE
last(T) INSET AUTO_HIGH [^]	last(T) NEQ ((WEIGHT*(RULE_SCORE_EXP+LOOKUP_SCORE))) [^]	ERROR
	last(T) = (RULE_SCORE > MAX_SCORE)	AUTO_HIGH_INDICATOR
	last(T) NEQ (AUTO_HIGH) [^]	ERROR

Table 5.28: Entity Max Attribute Value

	T	O(T)
last(T) INSET ENTITYATTRIBUTEMAX ^c	last(T) = (CDD_RULE_ATTR_MAX) last(T) NEQ ((CDD_RULE_ATTR_MAX)) ^c	ENTITYATTRIBUTEMAX ERROR

Table 5.29: Entity Category Aggregate Score Logic

T		O(T)
last(T) INSET ENTITYSCORE [^]	last(T) = (SUM_MAX_SCORE # 0, $\sum_{i=1}^N RULE_SCORE_i * WEIGHT / SUM_MAX_SCORE * 100$)	ENTITYSCORE
last(T) INSET AUTO_HIGH [^]	last(T) # (SUM_MAX_SCORE # 0, $\sum_{i=1}^N RULE_SCORE_i * WEIGHT / SUM_MAX_SCORE * 100$) last(T) # (AUTO_HIGH_INDICATOR)	ERROR
last(T) INSET AUTO_HIGH [^]	last(T) # (AUTO_HIGH_INDICATOR)	AUTO_HIGH_INDICATOR
last(T) INSET AUTO_HIGH [^]	last(T) # (AUTO_HIGH_INDICATOR)	ERROR

Table 5.30: Output Function Product Group Rule Score Function Template

	T	O(T)
last(T) INSET RULE_SCORE^	last(T) = (WEIGHT*(RULE_SCORE_EXP+LOOKUP_SCORE)) last(T) NEQ ((WEIGHT*(RULE_SCORE_EXP+LOOKUP_SCORE)))^	RULE_SCORE ERROR
last(T) INSET AUTO_HIGH^	last(T) = (RULE_SCORE > MAX_SCORE) last(T) NEQ (AUTO_HIGH)^	AUTO_HIGH_INDICATOR ERROR

Table 5.31: Product Max Attribute Value

T		
last(T) INSET PRODUCTATTRIBUTEMAX	last(T) = (CDD_RULE_ATTR_MAX) last(T) NEQ ((CDD_RULE_ATTR_MAX))	O(T) PRODUCTATTRIBUTEMAX ERROR

Table 5.32: Product Category Aggregate Score Logic

	T	O(T)
last(T) INSET PRODUCTS [^] SCORE	$\text{last}(T) = (\text{SUM_MAX_SCORE} \neq 0, \sum_{i=1}^n \text{RULE_SCORE}_i * \text{WEIGHT} / \text{SUM_MAX_SCORE} * 100)$ $\text{last}(T) \neq (\text{SUM_MAX_SCORE} \neq 0, \sum_{i=1}^n \text{RULE_SCORE}_i * \text{WEIGHT} / \text{SUM_MAX_SCORE} * 100)^{\wedge}$	PRODUCTSCORE ERROR
last(T) INSET AUTO_HIGH [^]	$\text{last}(T) = (\text{AUTO_HIGH_INDICATOR})$ $\text{last}(T) \neq (\text{AUTO_HIGH_INDICATOR})^{\wedge}$	AUTO_HIGH_INDICATOR ERROR

Table 5.33: Final Output Function

T		O(T)
last(T) INSET FINALSCORE [*]	last(T) = (ENTITYSCORE + COUNTRYScore + PRODUCTScore)	FINALSCORE
last(T) INSET AUTO_HIGH [*]	last(T) NEQ (ENTITYSCORE + COUNTRYScore + PRODUCTScore) [*] last(T) = (AUTO_HIGH_INDICATOR)	ERROR
	last(T) NEQ (AUTO_HIGH_INDICATOR) [*]	AUTO_HIGH_INDICATOR ERROR

5.4.7 Reading the TFM

In order to understand this representation a little more clearly it helps to utilize an example. Consider a fictional bank, MEGABANK, that has several million clients. MEGABANK is required by its regulators to collect information as it on-boards its new customers and to monitor existing ones for potential indicators of money laundering related risk. When the client John Doe sits with a relationship manager they are asked many compliance related questions so that the bank can manage the risk of doing business with John Doe. The questions asked add values for the fields we collect in Table 5.4.3 for John Doe's client record. Table 5.4.3 gives a holistic view of all the assets that should be collected for each category and aligns directly against EU (2012). In a situation where the EU adds a new data point to be collected the person responsible for this TFM representation would update the appropriate category.

While MEGABANK could use manual processes to check the risk that business with John Doe gives them and assign a risk band accordingly, it is very tedious to do so in a bank with millions of customers. As people change roles, move homes etc. it is incumbent upon MEGABANK to track changes in case it introduces a change that causes a material impact on the risk band John Doe is assigned to. Technology simplifies this effort significantly. In the case of the SAS model it scores data presented from the client records, like John Doe, against information repositories that hold data on sanctioned countries, known terrorists etc. SAS made a development decision to separate the model into a number of groupings to simplify the maintenance of the code base. This on examination is a good decision in terms of implementation. It is not reflective though of how the industry at large documents models and is definitely an example of a very transparent industry solution.

In the case of our TFM model we align to the implementation decision to separate the rules into groups that thus become programs that are responsible for distinct inputs and outputs as presented in Table 5.4.2 focusing on country, entity and products. The outlier being the programs responsible for aggregation and mechanical activity within the program logic. Each of which have subcomponents, Table 5.4.4, to help generate the scores they are responsible for. We can easily extrapolate the bounding logic of

the maximum and minimum values we saw in the code using these two sections of the TFM which ensure no vital information is lost. MEGABANK would have unique values for its implementation and as such the values for each of the variables presented in Table 5.4.2 would be a decision that MEGABANK would make in its deployment of the solution.

The same is true of the auxiliary functions presented in Table 5.4.4 as MEGABANK would make a decision based on the business it conducts and the type of clients. The SAS code provides a holistic reference but by using the TFM MEGABANK would be able to create an accurate map of their instance of that solution to provide to regulators. MEGABANK would be able to show its regulators how John Doe's risk score is created by starting with the output function of the TFM; Table 5.4.6.1 and showing the messages this creates; Table 5.4.6.

Assuming that John Doe is a high risk customer with a score of 900. The regulator should be able to trace using the TFM what data fields were collected. See how John Doe's score climbed. In this case we suggest because he was doing business with North Korea selling arms so that would flag high scores in both *Country* category rules and in *Entity* category rules. This would also flag the AUTO_HIGH_INDICATOR and immediately require a review by investigators into the dealings of John Doe.

This example is reflective of what a regulator needs to ensure a bank monitors and executes against. It shows the valid paths and paths that lead to errors being introduced. The regulator would look at all available documentation on a solution, examine the model, rules and variables. Often this event takes weeks of audit. The TFM could provide significant lift as it acts as a reference document that the regulator could train its staff to read and validate. However, some vendors have opted for code level transparency and while regulators may have some technical acumen such skill is not common. While we argue here the TFM is readable we validate this assertion later in this thesis. Ultimately, regulators need documentation they can read and understand. Complex training requirements would cause problems. The fact that the TFM is language agnostic would help as often localisation issues are introduced in solution documentation where it is available resulting in further effort to translate accurately.

5.5 TFM Analysis

Financial crime detection systems are an example of modern complex programming applications. Earlier in this thesis we defined financial crime detection systems as critical as they have a profound effect within our society to root out illicit behaviour and bad actors. The concern with such systems is that with such breadth they can become incomprehensible and as a result would have the potential to not meet the specifications laid out by regulators.

Within the FATF guidelines FATF (2012) there is a very exacting set of information on how to conduct AML compliance at a government level and at an operational level in banking institutions. This acts as a regulatory reference point for defining the inclusions and exclusions for any model focused on regulatory compliance. In the case of customer due diligence one can note that the inclusions are focused on the entity, be that a legal entity or a human client. However, as with any set of laws there is a certain amount of interpretation applied in the implementation of those regulations into software.

The software models used in this case would have to use a series of disparate data stores in order to aggregate each of the specified elements and as a result of the risk-based approach that the EU demands EU (2012) there can be further variance associated with the priority of the elements to the aggregate risk score. By using the TFM we have a means of creating a complete and accurate documented representation of the model. The TFM contrasts well against other specification methods as it allows us to create a complete representation and allows for complexity through its use of partial functions in its mathematical logic. However, having direct access to the code means that we have a level of depth that would not be available to regulators and banks as the company may only provide higher level documentation that does not disclose this degree of transparency.

One of the major questions in this research has been to understand if the TFM has applicability across the board to describe an example of financial crime detection code. This work has sought to understand all the major approaches used by these systems in

order to build a comprehensive picture of where the TFM could be used successfully. With regard to our core use case we can see some coverage for the specification of the customer due diligence model by simply parsing the set elements to ensure inclusion. Full coverage testing is necessary to ensure that complete coverage is provided by the TFM specification.

As the entire model can be summarized into a series of concise artefacts we can observe and validate the compliance of the model with national regulations EU (2012) as we have been trained to read this form of representation. Without said training a developer could probably ascertain more value from the code directly but as previously pointed out clients may not get access to the code being utilized. As all data elements exposed are provided in clean and concise representations. Not only does this give us heightened clarity versus other specifications methods like TAM for example but it also allows for flexibility in the specification. As the regulatory entities themselves are very dynamic any programs implementing them and specifications therein should equally be as flexible for additions and subtraction of elements.

Empirically we can prove this by supposition of a new regulation being introduced. New regulations often require that the banks acquire and monitor a new data field As each category is similar in mechanics it does not matter where the addition occurs. Effectively the changes necessary would require the addition of the new rule in the programs should it be massively different in structure to the other data attributes, but assuming that it is not the only real addition necessary to the TFM would be the addition of the attribute to the set elements under examination. Now in comparison to other formats of reference material, even code, there would be a larger material change required. Aggregate changes or new category groups would have a similar functional change but would be more involved than just a specific data attribute but not dramatically so.

Further, the question of readability is answered through the specification as the regulators can quickly ascertain compliance through a swift analysis of the documentation which would be a boon for developers. However, this assumes that regulators would take the time to acquire the necessary expertise to read this form of specification.

As the TFM is built around messages and documenting each message type through a table our description of the above model is easily adjustable to include more elements as long as it doesn't impact the overarching behaviour of the model. As an example the proposed changes for the 5th EU anti-money laundering directive include several new areas of interest that would just be simple program additions to this specification. As this specification is event based distinct modules for each new regulatory change that extend the specification could be defined. Like the industrial example posed in Quinn (2007) this application is equally as extensible as long as functional impact is assessed as remaining intact.

The major influencing factor in this method is our ability to verify the output using verification based testing methods. As discussed in Jing (2000) there exist a series of different mechanized mathematical tools that have the potential to be used as a verification engine for specifications in this format. As legislation changes, the specification can be updated by adding new rows to the TFM representation per legal update and potentially could be checked using the methods outlined in Jing (2000). Given the sensitivity of the domain area it is difficult to acquire data from a financial institution for this purpose and thus was de-scoped.

This final specification could provide significant lift to industrial partners as it would improve the confidence associated with the specification of the program. We survey some industry professionals to validate this assertion in a later chapter. As a financial institution faces fines that can exceed millions and billions when they are found to be non-compliant with regulation there is significant pressure on vendors to meet the exacting demands of the regulations being represented in software form. The TFM helps by providing a specification of the model that is compact, has more clarity than traditional documentation standards, and is mathematically rigorous. The latter allows vendors to verify with a high degree of credibility that the program meets with the client requirements.

Some noted disadvantages to the method include the ramp up time associated with educating model developers with the approach. There do exist some tools that could be utilized such as Group (1997), Kahl (2003), Heitmeyer et al. (2005), Lawford et al.

(2004) and Eles and Lawford (2011), however none of these are natural extensions to more common development environments. For the most part they demand a separate kernel and installation which could lead to duplication of specification or documentation effort for the developer. Any such duplication of effort would quickly negate the benefits of the specification method as too much time would be wasted on formalizing the specification.

Another concern was the relative weak adoption of the approach outside of some research centres globally. However, as we've seen through our analysis of the other specifications many formal methods face a similar issue in terms of adoption to industry. Some more recent examples like Singh et al. (2017) show continued application of tabular expression technology to describe complex programs. Should more tooling around the approach exist in more modern integrated development environments the potential for more mainstream adoption exists.

A further concern is the realistic lift of the method versus good development practices where code governance and documentation standards are in place. Best practices as seen in this industrial example allowed for the CDD model to be flexible in multiple modes so that attributes and categories could be manipulated with ease for a bank. Banks have unique tolerances and risk appetites which demands such flexibility in their regulatory compliance tools. The TFM becomes very repetitive in this case and ultimately doesn't provide much value in the face of such aggregation elements.

The TFM has benefits but there are still concerns on whether they outweigh the disadvantages above. An experienced developer that has been educated in formal methods would quickly acclimatise to using the method to address new regulations and extend the model program as appropriate. This efficiency gain coupled with the verifiability of the tabular representation provides developers in this domain with enough justification to warrant exploration in other financial crime applications.

However a developer inexperienced in formal method approaches would be critical of the time delay to generate these specifications but would probably appreciate the insulation they would provide for regulatory risk. Further, the industrial code presented

while a real world example is only one type of operational system that would need such specifications and other systems would introduce further complexity in the TFM representations accordingly. Given the lack of broad industrial application of these approaches it is questionable if developers would truly see advantage in using this approach without the presence of a mandate from their marketplace and management chain.

We explore some strategies to potentially test the coverage of the specification provided above and to validate the ease of comprehension through a survey in the next chapter of this work.

5.6 Summary

This chapter introduced the reader to an industrial application of financial crime detection methods in the form of a customer due diligence model. A customer due diligence model is an approach to risk rate both traditional clients and legal entities. Risk rating takes into account many variables describing the behaviour of the entity. These variables can be more or less important to a financial institution depending on its active business model and risk appetite.

An example model tied to the subject area was introduced and described literally in terms of its intent and the code based representation. All aspects of the model were discussed including aggregation, rule logic and variables considered in the operation of the program. To add clarity each aspect of the model was divided into subcomponents addressing the areas of interest documented in EU legislation (EU, 2012).

The model was described formally using the formal methods Trace Function Method approach (Parnas and Dragomiroiu, 2006). This broke the program into a series of discrete elements finalizing in an output function in tabular form. This tabular form is easy to read to those trained in its use, verify and allows for efficient consumption of the logic of the model. An analysis on the advantages and disadvantages followed arriving at the conclusion that the method provides some lift in financial crime detection system

design whilst having concern about the ramp up time for education and the lack of integration into traditional development environments.

"Quality means doing it right even when no one is looking."

Henry Ford



Testing & Discussion

6.1 Introduction

To ensure the validity of the approach it is important to test the specification we generated as part of the TFM process. We examine two fundamental questions in this chapter;

1. Does the specification provide adequate *coverage* in accurately describing the requirements for the software?
2. Does the specification provide lift in *comprehensibility* of complex software requirements?

In order to answer these questions we attempt to solve using both quantitative and qualitative approaches. For the first question we examine some systematic testing strategies that could be leverage to validate the coverage of the specification. We postulate whether we could utilize an automated means using a mechanized mathematical tool for this purpose as has been utilized in the past research (Jing, 2000).

In terms of comprehensibility of the specification we are reliant on human interpretation of the specification and as such generated a survey to study the perception domain experts of the TFM approach around requirement description. This study provides examples of specifications using the TFM and requires experts to comment on ease of use, accuracy, detail retention and flexibility. We analyse and discuss the results of the survey and draw conclusions regarding the usefulness of the method in this space.

6.2 Coverage Testing

Independent of the domain problem being solved it is important that any software module undergoes testing. The IEEE standard for software test documentation IEEE Std 829-1998 defines testing as "the process of analysing a software item to detect the differences between existing and required conditions (that is, bugs) and to evaluate the features of the software item." The inference from this definition is that the software module under test is presented with all possible variants of input variable feasible for the program and that a verification of the expected outputs for these variables is performed.

Given the complexity of modern programs, some of which utilize millions of lines of code, it can be difficult to create an exhaustive suite of tests to examine all variations of input variables for a given program. As such, common sense needs to be applied in determining bounding criteria for the scope of testing that needs to be performed and to leverage as much automation as possible to increase the coverage of these test suites across the software. Coverage testing tools, like those discussed in Horgan et al. (1994), would provide significant lift in helping developers automate much of this process as we recommend.

As we are utilizing an industrial example for our use case we are going to assume that the conventional forms of testing have occurred and rather focus instead on the verification of the specification generated over the course of this research. Pivotaly, we will examine coverage testing the specification itself. Quinn (2007) defines requirements coverage testing as being the examination of the component to verify that it adheres to the specified requirements. They continue by noting that the aim of coverage testing is to cover as much of the description as possible so that the implementation has been exercised and shown to be consistent with the description and no assumptions made regarding frequency.

The results of applying coverage testing to our TFM specification would show which aspects of the description were accurately satisfied. Quinn (2007) also discusses limitations associated with the approach. They note that no weighting is assumed for any part of the description as we assume equal probability for all the requirements. Given our previous remarks earlier in the chapter around the increasing complexity of modern software programs testing every potential input demands an ever increasing

volume of test cases. Arguably, we should be able to use a testing harness like those mentioned in Yang et al. (2009) which examines many examples of these tools for configuration, automation and test reporting functionality but given the unique nature of our descriptions there would potentially be further development necessary to extend these.

In the case of our use case there are a broad range of potential values that could be utilized in the input variables consumed due to breadth of the domain problem. Example data would be inclusive of elements of names, date of birth information, address information etc. As such we can only apply a limited range of values and permutations to test the accuracy of the specification giving rise to the potential for other bugs to arise later in the implementation.

The TFM description of the requirements does provide some lift when conducting coverage testing. Namely;

1. Due to the fact TFM representations are tabular they simplify coverage testing execution by each row of the table partitioning the inputs of the module. The aggregate of all partitions thus becomes a simplified variant showing full coverage of the requirements description.
2. For each partition, a corresponding testing pattern can be inferred. Quinn (2007) notes that these would include sets of events that are expected at each point in a trace. From these, the possible sets of test case traces can be derived for each row of the table.
3. By using these test patterns it is possible to generate test cases randomly or with a predetermined selection of constraints. Given this option it is feasible to generate multiple test cases per row of the TFM description.

Earlier in this thesis we discussed some of the previous work conducted in the area of tabular expressions such as Jing (2000), Lawford et al. (2004), Heitmeyer et al. (2005) and Eles and Lawford (2011). Many of these attempted to utilize automated means of checking tabular expressions using a mechanized mathematical tool. Many ended up using an automated theorem proving system.

Automated theorem proving is a field of automated reasoning which concentrates on the proving of mathematical theorems using a computer program. A theorem is

made up of two main components, a set of assumptions and conclusion derived from the said assumptions and a set of inference rules. In computer science we can describe and document certain aspect of software using mathematics. These descriptions can be transformed into theorems describing there components, a theorem which then can be verified by using automated or human means. Using this technique we can draw certain conclusions about the validity of a specification for a piece of software.

In Janicki (1995) we are introduced to the concept of describing functions using tabular expressions. Earlier work provided us with a logic for comprehending these expressions Janicki et al. (1996) and gave a foundation for us to see that by using such means of documentation it made the possibility of verification easier. Tabular expressions, fundamentally, are nothing more than a different format for observing the components of a function and the resultants for operations of that function under varying conditions. Using these components it is then possible to construct a theorem, based on the logic discussed in Parnas (1993a), and to verify whether we can or cannot verify this theorem.

This aligns with our goal of checking the coverage of our specifications and provides a means of cleanly automating the tests. In the previous iterations of research into a checking tool PVS Owre et al. (1992) was the tool chosen for this purpose. PVS has a number of built-in structures that support the checking of tabular expressions such as its built-in table data structure as well as automation in the form of a batch mode Owre (1997). It also supports partial functions and the logic proposed in Parnas (1993a) which is crucial for automating the checking of tabular expressions. PVS is also the only theorem prover known to successfully answer the challenge set in Parnas (1993b).

Given the fact that our use case is effectively an aggregate model automation to the degree described in Jing (2000) is unnecessary. However, in future research of more complex financial crime programs such automation could provide significant value in shortening the validation cycle for developers working in the field. This would be especially true if we were in a position to test against real data extracted from a financial institution as the volumes in question would be too high to allow for manual review in a reasonable time frame.

6.2.1 Testing Strategies

As our use case is an aggregate model made up of several subcomponents we can dissect the model and deal with each subcomponent in isolation should we so desire or look at the final outputs and test there. Given that a customer due diligence model is designed to scale horizontally to map to changes in legislation it is more beneficial to create strategies based on both of these perspectives and determine the test cases necessary for coverage of the requirements considered.

However, it would be foolish to assume there is only one strategy to test for adequate coverage even given the domain regulatory requirements. Cai and Lyu (2005) makes solid arguments for having several testing profiles or strategies to solve for coverage to ensure against potential faults within the software.

We consider the following testing strategies building on previous research that also examined industrial applications for TFM use cases (Quinn, 2007);

1. Strategy 1: Consider every test pattern respectively and for all successive elements of the trace choose a message. This set of selected messages should not, in accordance to the description, result in an error state.
2. Strategy 2: Consider every message and for all successive elements of the trace sample at least one test case which should undergo multiple test passes. This test evaluates for constructive messages and error states.
3. Strategy 3: Consider all test patterns and evaluate that every combination of constructive and error state messages. Assuming only test cases not resulting in a member of the error set are considered this should attempt to exhaust all constructive message outcomes.

Test Strategy 1 The purpose of this strategy is to look at evaluating the constructive traces for the software and ensuring that the outputs don't result in an error state that wasn't included in the trace. Whilst it's prudent to be exhaustive in testing this given the domain of the problem combining multiple data elements for the model there is still the potential for error to creep in. This effectively results in more of a sample data set applied to the test pattern. The description in the trace tells us that we shouldn't

arrive in an error state but without trying the combinations of all messages it's difficult to give a complete guarantee.

An example from our industrial use case to consider would be industrial occupation codes, *OCCUPATION*, (Commision, 2008) an element of the set *EntityScore* 5.18). For this strategy one may consider one type of *OCCUPATION* listed in Commision (2008) and on evaluation deign the test pattern a success.

Test Strategy 2 This strategy ensures that more comprehensive coverage compared to that of our first pattern, by making sure that every message occurs once. It attempts to give a broader test list of all potential inputs for the program. This mechanism provides a more controlled means of validating the interfaces of the model. While still not exhaustive it gives a better indication of potential bugs and error states within the software. Failures detected though could also be indicative of issues in how the description was implemented rather than the description itself.

Continuing our example from our industrial use case for this strategy, consider the variable *OCCUPATION* from 5.18 once more. We would sample several *OCCUPATION* types listed in Commision (2008) along with a number of error candidates and on evaluation deign the test pattern a success should it consistently react as per the description.

Test Strategy 3 This strategy is similar to the first test strategy in that attempts to test every variation of messages but deviates in that it is inclusive of those that result end in an error condition also. This is the most exhaustive test strategy and requires a huge number of variant test cases cascading through the model. It also would be the most comforting in terms of ensuring a fully comprehensive result regarding the accuracy of the description as it would showcase all potential failures and unpredicted events within the system.

As before we can continue our previous example from our industrial use case to explain this strategy. Once again considering the variable *OCCUPATION* from 5.18. We would execute all six hundred and forty *OCCUPATION* types listed in Commision (2008), approximately, along with a number of error candidates and on evaluation deign the test pattern a success should it consistently react as per the description.

Recommendation Given the regulatory nature of the domain use case we believe that only the most exhaustive and comprehensive test strategy would satisfy regulators. As such test strategy 3 would be the logical path to utilize for financial crime use cases pertinent to the anti-money laundering domain. Whilst this does mean significant more pressure on the developers to ensure accuracy they can ensure the confidence of their clients and subsequently regulators because of this rigour.

Further, it would be our recommendation to maximize the use of automation to fulfil this requirement as it would become very tedious to manually execute each test case.

6.2.2 Analysis

Fundamentally, we arrive back at the question of whether the specification is complete. Given the strategy necessary to ensure that completeness requires extensive levels of information available to a bank it is difficult to give clear guidance on this. Our argument is that the TFM as a method should be able to provide the right degree of rigour and is expressive in its construction to allow for mathematically accurate representations of industrial applications.

Further, we do not have a suite of tools to automate the test strategy we recommend above which necessitates a large manual effort to ensure completeness. Thus, we return to our assumptions that if we are confident that the TFM is mathematically sound and we are further confident that the representation of the CDD model is an accurate and correct specification, arguments can be made that the specification is somewhat complete. Only when we have conducted full testing of the model TFM specifications via a test harness with complete bank data would we be able to be completely confident in the completeness of the specification.

6.3 Comprehensibility Testing

In order to effectively evaluate comprehensibility of the method it is necessary to involve experts that have had exposure to financial crime software solutions to evaluate

the method. Such endeavours are qualitative in nature but to ensure a more robust evaluation we prepared a survey which was circulated to several unbiased experts in the area. To ensure a more accurate assessment we included several important organizational roles pertinent to the software life cycle of a financial crime solution, namely;

- Software Developers
- Data Scientists
- Product Managers
- Domain Experts

We make a series of assumptions regarding this study. Firstly, we assume that each of the respondents have limited or no exposure to formal methods and specifically the Trace Function Method Parnas and Dragomiroiu (2006). Second, we assume that each of the respondents plays is active in presenting software solutions for financial crime domain use cases to financial institutions and thus, are exposed to the limitations, grievances and gaps associated with this time of software. Finally, we assume that our experts have an understanding of mathematics and understand the software life cycle. This is critical to ensure that they have the potential to see lift in a more formal approach to writing software document specifications.

6.3.1 Survey Structure

The survey firstly introduces the reader to the importance of both readable and precise documentation. This is accomplished by guiding the reader through a guided example that showcases the issue of separating readability at the cost of precision and precision at the cost of readability.

Building on this foreword we introduce the trace function method and highlight the potential lift that specifications that are both easily understandable/readable and precise. The justification for this basis is provided by highlighting the concerns of financial institutions in being able to provide accurate descriptions of software utilized in anti-money laundering compliance. We can assert that regulators of the banking sector are not experts in software development nor formal methods. They are our

ultimate reader and thus the representations need to be presented in a format that is understandable by those that have had limited exposure to these topics.

The survey itself is provided as an appendix to this work and can be found in **Appendix 2**. This survey was sent to ten industry professionals and of the ten we received responses from five. These five provided their comments through a series of expert interviews.

6.3.2 Results

The results of the survey echo a similar study into formal method utilization in industry, Barnes (2011). There the author concluded that the challenges that exist for formal methods in industry could be summed in the following categories;

- scalable
- notation approachable to all stakeholders
- expressive (ease of capturing the problem)
- tool supported

In the case of the TFM arguments can be made for each of these problems given the availability of supporting literature examining industrial examples, some tools have been developed in different academic settings and the constant demonstrations by the methods author that the method is easily trained to developers.

However, even making that argument there are concerns of findings similar to Wassing and Lawford (2003) in another formal methods project with industry. We looked at elements of this project earlier where some tools were created for the purpose of supporting the method namely Lawford et al. (2004). In this case the authors noted that the use of formal methods in the project was a success but the reality is that a lot of this success was built off the level of expertise of the team involved whom were highly trained in this method. Further, they noted that the industrial client in this case, a nuclear power plant, had sufficient motivation to ensure the success and thus was able to bring resources to bear to ensure success.

The author also critically pointed out concerns regarding the practicality of using formal methods. Some issues they noted were not completely solved like, timing but were mitigated during the course of the project. The biggest critique though was the lack of comprehensive tool support for using formal methods on that project. A situation that frankly has not advanced much since this paper was authored.

While some tooling has been created in academic environments there still is no suite of tools available that would be necessary for an industrial environment. This was cause for concern amongst our survey respondents.

Candidly even the author of the TFM method has similar concerns. As discussed in Parnas (2010) the author queries why formal method approaches have not been adopted into mainstream software development. There are several key observations in this research that examine previous industrial use cases and if the formal method employed was further adopted. Generally speaking it is clear that all the industrial examples specified are isolated cases of use of the formal method employed pointing at major gaps between research and application.

Reasons for this include the divided perception of the use of mathematical specifications in software development. As software complexity continues to increase software developers have become jaded regarding the potential of employing mathematically based specification methods to improve software quality. Another cited gap is the foundational training according to Parnas (2010). Engineering principles taught to students of software development when formal methods were originally proposed are generally no longer as important as technology related topics. The final gap is a gap between the mathematics employed by software developers where the methods they use could be simplified by older mature methods once dominant in the field.

Extrapolating from these remarks it is clear to see that many formal methods seem to be trapped within a bubble and are extremely unlikely to be utilized in industrial software development companies. The concerns delivered in Parnas (2010) were echoed

by the survey respondents. For the survey a cross section of developers, domain experts, and product managers. The consensus of our survey results was that formal method utilization would not be beneficial in this industry.

The examples in the survey were understandable for those with a heavier mathematical background. One of our respondents, a senior developer and product owner noted that with regard to the examples provided; *"They give an outline of the main access methods as well as a concise and range limited description for their operation, with boundary conditions clearly indicated"*. Their focus area is *"Data storage and analytics platform for storing sensitive time series data that is immutable and is FDA CFR 22 Part 11 Compliant"* for compliance applications. The company in question is a small development house employing five to ten developers. However when questioned on the potential of using formal methods they responded by noting that they *"Considered using formal methods, but determined use would not be cost effective"*.

Further, the same respondent elaborated; *"Rigorous testing and compliance verification provides a greater certainty and validation of applications than the appliance of formal specification that would still need testing against the implementation. The formal methods would prove the design but not validate the implementation"*. This provides industrial validation for some of the concerns stated in Parnas (2010) and Barnes (2011). *"Compliance with regulatory requirements is always bespoke to the needs of the regulatory body and their testing paradigms, which required individual approaches per body"*, was the concern regarding using the approach in the regulatory compliance domain. The conclusion was quite stark; *"Math is always off-putting to the layman, which is a barrier to adoption"*.

Another respondent, a subject matter domain expert from a large software house with over one thousand developers, noted that *"The CDD example has the clear precision and description of the input and output but lacks the description of the process."* Their concern was that the utilization of mathematics obfuscated the meaning of the regulation and could cause difficulties for a regulator. The business focus of this respondent is *"Transaction monitoring, CDD, and sanction screening for AML compli-*

ance."As such, their opinion carries impact as they represent a true practitioner in the field.

They were more optimistic about utilizing the TFM (Parnas and Dragomiroiu, 2006) stating that *"The approach can further reduce the ongoing operating cost due to ease of software maintainability. In addition, the compliance risk can be reduce due to better model interpretability"*. They further noted that *"Clients expect to have end-to-end understanding of their own financial crime program. Ultimately, the software is deployed to support the program, so program owners such as CAMLO¹ has the ultimate responsibility to respond to regulatory requests"*. Extrapolating from these remarks shows that there is potential for use if the barriers regarding training and comprehension can be addressed.

Ultimately, the responses to the survey had a common theme regarding the high barrier to train developers to use the method, impart the knowledge around the mathematical notation and ensure that the rigour necessary to ensure document accuracy is ensured. Even with such provisions our respondents had doubts that the use of the TFM would be provide significant enough lift to warrant the expenditure necessary to train their teams. This was held true even when the concern regarding regulatory risk was weighed with them.

Potentially, when formal methods evolves to map to the engineers work-flow and intuitive mechanisms exist via non intrusive tool kits for application development there is potential for formal methods to help financial crime developers. Quite some time has passed since Parnas (2010) and it does not look like industry is ready for these approaches yet.

6.4 Summary

This chapter proposed some strategies to test the coverage of the TFM representation of the model provided. It focused at three levels of exhaustive testing and concluded that

¹CAMLO; Chief Anti-Money Laundering Officer, a senior role in banking responsible for a banks regulatory compliance framework.

for regulatory driven applications that only the most exhaustive would be acceptable to regulators.

The chapter further introduced a survey to check for the comprehension level required by developers and domain experts working in the financial crime industry when studying TFM specifications. There was some optimism about lift using the method to persuade clients of accurate software specification but the ultimate conclusion was that the barrier to entry was the heavy reliance on complex mathematics and training for developers. As such they did not see a place for formal methods in their development teams at present.

"Programming is like sex: one mistake and you're providing support for a lifetime."

Michael Sinz

7

Conclusion

7.1 Results

We consider the main results of this thesis are as follows

- A Trace Function Method Description of a Financial Crime Model focused on Customer Due Diligence
- An analysis of financial crime detection systems, with an in-depth review of how they attempt to systematically solve for financial crime activity coupled with some examples on how they do this in code.
- Some background reference material on regulation for financial crime detection software from the European Union and other regulatory entities.
- A review of popular formal methods technology and in depth analysis of the use of tabular expressions for document driven design.
- A survey sent to industry professionals regarding their ability to comprehend the method and to learn their opinion on whether it is industry applicable.

More detailed discussions of these results are considered later in this chapter.

7.1.1 Review of Formal Methods Technology and Introduction to the Trace Function Method

Chapter two provides a succinct review of the strengths and weaknesses of several popular formal methods utilized to describe critical systems. We evaluate several can-

didates and decide on the utilization of the trace function method in order to describe our industrial use case.

Further the chapter introduces the mathematics behind the trace function method and helps provide tangible visualizations for the reader so that they can understand how we formulate the tabular representations of the use case model later in the document. These preliminaries showcase the difference between the Parnas approach Parnas and Dragomiroiu (2006) and what it offers to software developers who are looking to create bullet-proof specifications.

Further, this chapter gives the reader several examples of how to create tabular expression trees from several table types, thereby allowing one to assess whether the program could easily be translated using this formal method approach.

7.1.2 Model Legal Reference Analysis

Anti-money laundering compliance is a thorough field with a broad depth of legislation adopted by governments globally. Customer Due Diligence is a process that exists within the broader area of this legislative area. Chapter 3 provides the background to the legalities adopted by the European Union member states EU (2012) that provide the basis for financial regulators efforts at a national level.

Our analysis shows how FATFFATF (2012) recommendations are considered and enacted. These in turn put further compulsory requirements on the financial services sector. Given the volume of transactions and the client populations it quickly becomes a problem to scale any manual process to search for money laundering indicators. Technology has been deployed across all financial service products and channels for this purpose but these technologies are often black boxes.

Given the urgency in government and the punitive measures adopted by regulators to banks that fail to comply further rigour is necessary within the technologies deployed to solve for money laundering use cases. Our analysis concludes that through better standards of verifiable documentation using formal methods software developers could meet that demand with ease. Further, it would provide them with an easily managed representation of all applied code elements and how it meets any legal reference.

7.1.3 An analysis of financial crime detection systems, with an in-depth review of how they attempt to systematically solve for financial crime activity

We examine financial crime detection systems in Chapter 4 as a modern example of a critical system. The basis being the integrity of the global financial system and the implications on terrorist financing. We examined what mathematically driven techniques are applied to capture illicit behaviour and conduct an analysis on how that would translate into use with a formal methods approach.

Each technique has mature use within the sector to solve for financial crime concerns. Industry uses several techniques in combination in order to address different vectors of the attack schemes that criminals use to target financial institutions. We examine the history of research into each technique to isolate their strengths and to understand how complex it would be to translate them into a formal method representation.

Example code was examined to determine how fit for purpose the TFM was for industrial level code. Two examples were discussed one which examined the problem of name recognition, a common concern from the regulations enacted throughout the European Union. Second, was a neural network applied to a specific seeded data set. This second example is more illustrative of modern financial crime techniques as they attempt to embrace more predicative capabilities coupled with machine learning.

7.1.4 Trace Function Method Description of a Financial Crime Model

This thesis presents a full described model of a customer due diligence problem in chapter 5. The initial code for the model is provided so that the reader can quickly assess how the trace function method transforms the code into a compact entity that can be readily assessed and verified (manually or through mechanized mathematical tools). The model integrates a complete interpretation of EU law and FATF recommendations on the area of KYC (Know Your Customer) data collection and risk scoring.

The model is a translation of SAS model code that specifically targets client risk rating problems. The model incorporates numerous data elements mostly harvested

from curated collections of information on known terrorists, sanctioned individuals and drug traffickers. It looks at risk topologies by dividing it into three pillars and creating a nested aggregation facility that ends with a final aggregate score of comprehensive customer due diligence related risk. Once that score is provided other operational systems use it for the purposes of ongoing surveillance of a given client such that the bank can make a quick determination should they become too risky to do business with necessitating a report to regulators and de-risking the client from their book of business.

7.1.5 Use Case Conclusions

Financial crime detection systems are a valuable area which we believe worthy of further analysis and applications in formal methods. The above work examined code constructs that attempt to solve for customer due diligence risk ratings within the financial crime domain.

We have earlier in this work discussed the plethora of different methods utilized in this domain to solve for financial crime. What is clear though is that formal methods could play a vital role in the definition of new scenarios and model developments. Currently considerable effort is expended by financial institutions and governments in efforts to optimize detection methods at play across the financial spectrum.

If we take a more formal approach like Parnas (2003a) there is more impetus put on designing a higher quality of specification which is more code or realization agnostic. This gives us many benefits in that it provides a larger degree of flexibility in how the realized design is applied to new technological advances. Further, it provides the detection methods themselves with a more refined definition and enhances an analysts knowledge of all potential outcomes in the implemented code base.

While very different in design from traditional applications for formal methods there is a clear bridge here between improved software design and a realized goal of better detection facilities impacting us all.

The criticality of these systems is not in dispute but due to the proprietary nature of many of these systems it is difficult to assert the impact of formal methods on a development cycle at one of these institutions. However given the predominance of agile

paradigms at this institutions one could foresee considerable push back on utilization of the approach.

7.1.6 Survey into Industrial application

Several strategies were presented with regard to solving for coverage tests for the CDD model and examining the representations provided. Of the three strategies posited only one could provide the right degree of rigour that would satisfy regulators and those under regulatory supervision for their software.

To further explore the comprehensibility of the methods presented a survey was issued to a group of industry domain experts, developers and product managers in the financial crime domain. The aim of the survey was to get a measure of how effective the Trace Function Method was at creating specifications that industry professionals could understand. A further goal of the survey was to understand if industry professionals could see a value in using formal methods in financial crime software development.

Of the 10 people surveyed, the respondents indicated a clear commentary on their lack of comprehension of the specifications provided. Even when give ancillary material to improve their education they found the specifications convoluted and complicated. After some discussions they agreed that some lift could be provided assuming regulators could be trained into understanding the specifications themselves but it would take a massive investment in education of developers, bankers and regulators at a global scale to be worth the cost of implementing formal methods at an industrial scale.

Further critique was given regarding the lack of industrial level tooling available to support the use of any formal method and sufficient reference sources from other industrial partners to eliminate concerns regarding the risk of deploying these methods first.

7.2 Critical Analysis

From the outset this research has been about proving that formal methods have a place in describing industrial financial crime programs. As argued in Parnas (2003a) formal methods provide a safety net enabling the verification of software programs and a useful mechanism to consume complex design elements. Within the context of our industrial domain, financial crime, a complex reference entity in the form of regional legislation exists to provide hard limits and precise operational demands of the financial services sector. EU (2012) is an exacting framework which demands EU member states to address through legislative measures within their own national constitutions. It specifically addresses the area of anti-money laundering and related topics under that umbrella and calls for operational means of inspection the data collected by financial institutions to highlight anti-money laundering risks.

This research examines the use case of customer due diligence which is a sub-component of anti-money laundering measures. Customer due diligence primarily asks banks to acquire from prospective and existing clients authentic documents to prove identity, source of wealth and to disclose potential risks. As per FATF guidelines (FATF, 2012) an accurate reflection of customer data is imperative to measuring the likelihood of money laundering related risk in a given financial institutions. Clients of significant risk will not disclose all information if possible and will hide behind familial entities, mules, shell corporations or simply just provide doctored information to bypass control mechanisms at the bank in question. As such, banks often turn to technology to solve for customer due diligence related concerns in order to ensure that they can prove they provide the correct degree of rigour on client risk profiles to regulators.

The intention of using technology in this space is to provide a safety net for financial institutions to conduct the analysis on the client base for them. Often this is driven by the volume of clients and transactions involved but also due to the complexity of modern financial channels and how clients interact with them. As such an implicit relationship exists between regulators, the banks and vendors in this space to meet the stated needs outlined in legislation. There is however, a burden of proof required by regulators to ensure everything is operating as it seems and when this fails the penalty

tends to be immense financial fines and threats to the banks license to operate. Vendors are thus placed in a position where they need to verify and validate inputs and outputs associated with their programs to ensure adequate coverage.

As a result of these explicit laws and the implicit relationships between entities this research argues that formal methods has a place to bridge the gap between software and legislation by providing accurate documentation elements that can be verified for accuracy. The field of formal methods is broad and there are many potential candidates for use in this area as proven by Abrial (1996), Heitmeyer (2001) and Lawford et al. (2004). This research argues that Parnas and Dragomiroiu (2006) represents a logical choice of candidate method. The argument is that the TFM represents a modern take on methods employed for almost half a century in both academia and industry stemming from original work introduced in Parnas (1972*b*). Extensive research has been compiled across several organizations focused on proving precursors of the method, the method itself and developing tools for use with these methods. It provides a mathematical model that provides a high degree of flexibility allowing for the representation of highly complex programs as clearly illustrated by Bane et al. (2004).

The contrary view is that the method provides too much functionality for a use case that can effectively be described as a aggregation model. The use case examined here is just one aspect of anti-money laundering programs and not all are simple aggregation mechanisms. Further, as you attempt to describe more complex programs that utilize more complex techniques the full breadth of the TFM would provide support without the need for extending the method. Another concern is the complexity of the traces associated with the output of the method and the usability of these traces. Given that these traces are machine readable for systems like PVS, a theorem proving system developed as per Shankar et al. (1993), that can be verified and have successfully been analysed in the past in Rushby and Srivas (1993). As per machine readability we argue that simple tabular representations are a more effective means of consuming complex data relationships than relying on text that may be interpreted in several ways.

Other concerns point to the lack of industrial penetration by this and the other formal methods and build upon the argument presented in Parnas (2010). Without substantive

changes to the educating engineers on the utility of formal methods and the lift they provide it is unlikely that any industry, financial crime software development included, would use any of them to pursue a more comprehensibly accurate form of specification reference documents. This is especially true given the lack of tools available to developers to support these approaches.

This research posits that it is feasible to use formal methods to describe financial crime software programs. Further, it argues that the benefits of these methods is higher transparency and a white box approach thus improving the confidence in such systems by financial institutions and those that regulate them. The TFM represents a formal method that could be used to provide the coverage needed to describe such programs as it provides the mathematical means of doing so and the flexibility to describe complex critical systems. However, as stated given the lack of penetration of formal methods into software development it is extremely unlikely that these approaches will find a use in industry at present, but potentially in the future.

7.3 Future Work

This work examined an entirely new area for formal method based design specifications that of financial crime detection that to date has been overlooked academically. We have only begun to research this area and did not consider some of the broader areas of interest in this field like neural networks, machine learning or Bayesian/Predictive Analytics. As these areas were out of scope for this research we would like to advance our research by completing a review of models and coding standards for design using these other techniques.

A logical next step would be the design of a means of verifying such descriptions to allow developers of financial crime systems a mechanism to automate the verification of the expressions generated during development. Work such as the research in the area like Eles and Lawford (2011) makes more sense as it applies the tabular expression checking context and would act as a foundation to such efforts. It does introduce some potential logical gaps but ultimately provides more usability. As such

we would seek to build a more cohesive integrated framework with mechanized mathematical tools for verification purposes but utilize the logic of Bane et al. (2004) for completeness.

As our specification design is intended to work with the financial crime detection domain another area of future work would be the extension of an integrated tabular expression environment with development tools designed around specifying these systems. While it is not feasible to integrate cleanly with proprietary vendor offerings it would be of interest to see if we could construct a tabular development environment with specific emphasis on modelling languages like SAS, Python or R. This could solve for some of the concerns raised by our survey respondents.

One limitation that was a major concern in our research was the scale and complexity of the information being examined. Modern financial crime systems are designed to operate on tens to hundreds of millions of financial and non financial transactions daily. This means for a pretty harsh but data rich environment where many permutations and variances on the data can exist. Testing new scenarios described in our tabular specifications here would hit scaling problems. It would be interesting to conduct further research into a scalable tool suites for formal methods that examine proving strategies in an environment like Spark and Hadoop which are typically used in financial institutions.

7.4 Summary

This work has aimed to provide a basis to evaluate a formal method description approach using an industry use case so as to understand if the method provides utility to developers. By employing an example of a new critical system in the financial crime detection domain it was hoped to expose new applications for both a formal methods based approach to document driven design (Parnas, 2003a) and further to provide developers of solutions in that field with a mechanism to check these specifications.

Methods utilized in financial crime detection and the back-end regulations were discussed and insights provided. An examination of code used in this area yield an analysis that suggests that tabular expressions based on financial crime detection code provide a more compact form that also could be verified. Further we postulated that

formal methods would give great value to this domain but would be hindered by usability and currently adopted development paradigms by the vendors that dominate this field. In summary, it is clear that while formal methods provide some lift it is not enough at present to persuade modern software houses to adopt them for solving industrial grade problems.

8

Appendix 1

8.1 CDD Use Case SAS Code

8.1.0.1 CDDR_C101

The following code utilizes several variables and is focused on a data field examining the client field 'ORGANIZATION'. The code works by passing the variables to the execution harness. In this case the ORGANIZATION registration information is being evaluated against a country watch-list. As such the attribute in question, REGISTRATION, is evaluated by the model. Some of the variables manage ceiling, floor and auto setting values for situations where the bank would require those values to be preset. It has logic to evaluate the score of this rule versus the maximum allowable score for a rule within a category. Further, logic is provided on value range controls and the setting of the auto_high_indicator tracking flag.

```
%cdd_rule_attr_level( attribute_code=&attr_name
, party_type_desc='ORGANIZATION'
, weight= &weight
, max_score=&max_score
, min_score=&min_score
, auto_high_score=
, rule_score_exp=LOOKUP_SCORE
, not_exceed_range_YN=Y
, auto_high_exp=%nrquote(ifc(RULE_SCORE > MAX_SCORE,'Y','N'))
, triggering_value_var=ATTRIBUTE_CHAR_VALUE
, triggering_value_type_code=CHAR
```

```
);
```

8.1.0.2 CDDR_C102

This element examines the mailing address associated with the entity to determine if it maps to addresses provided in published sanction watch lists (OFAC, United Nations etc.) or with internal hot lists. 8.1, follows the same pattern as 5.5 in terms of input but instead examines a different aspect of the customer record. Each of the attributes examined under the input variable ATTR_NAME align directly to fields of interest in EU (2012) that banks are required to monitor.

Table 8.1: C102 Parameters

Name	Type	Description	Value
ATTR_NAME	Character List	Attribute Name	("STREET" "MAILING")
MAX_SCORE	Numeric Constant	Maximum Score	200
MIN_SCORE	Numeric Constant	Minimum Score	0
WEIGHT	Numeric Constant	Weight	1

```
/* Non-Personal Country Rule: Primary Business Address */

%cdd_rule_attr_level( attribute_code=&attr_name
, party_type_desc=' ORGANIZATION'
, weight=&weight
, max_score=&max_score
, min_score=&min_score
, auto_high_score=
, rule_score_exp=LOOKUP_SCORE
, not_exceed_range_YN=Y
, auto_high_exp=%nrbquote( ifc( RULE_SCORE > MAX_SCORE, 'Y', 'N'
) )
, triggering_value_var=ATTRIBUTE_CHAR_VALUE
, triggering_value_type_code=CHAR
);
```

8.1.0.3 CDDR_C103

This grouping examines the address of a potential beneficial owner of the account. A beneficial owner is someone that uses an intermediary in order to mask their activities. There can exist multiple layers in between the account and the ultimate beneficial owner. The practice of layering is a common tool in money laundering and can be commonly observed in use by drug trafficking cartels. 8.2 shows here the attribute of interest is focused on beneficial ownership risks and understand how much in terms of percentages is substantive risk for the bank.

Table 8.2: C103 Parameters

Name	Type	Description	Value
ATTR_NAME	Character List	Attribute Name	'BENE_OWNER_RESIDENCE'
WEIGHT	Numeric Constant	Weight	0.5
MIN_SCORE	Numeric Constant	Minimum Score	0
MAX_SCORE	Numeric Constant	Maximum Score	200
PCT_OWNERSHIP	Numeric Constant	Minimum Amount of Ownership Required	0.25

```

/* Non-Personal Country: Residence of Beneficial Owners */
%cdd_rule_attr_level( attribute_code=&attr_name
, party_type_desc='ORGANIZATION'
, filter_exp=%nrquote(related_attribute_type_code="OWNER_PCT"
and related_attribute_num_value > &pct_ownership)
, weight=&weight
, max_score=&max_score
, min_score=&min_score
, auto_high_score=
, rule_score_exp=LOOKUP_SCORE
, not_exceed_range_YN=Y
, auto_high_exp=%nrquote(ifc(RULE_SCORE > MAX_SCORE, 'Y', 'N'
))
, triggering_value_var=ATTRIBUTE_CHAR_VALUE
, triggering_value_type_code=CHAR
);

```

8.1.0.4 CDDR_C104

This field, 8.3, examines address information associated with directorships. A common practice to clean money is to utilize income/dividend splitting between directors of a shell corporation. Most national registrars of incorporation require that some KYC information is collected regarding directors but this for the most part is incomplete. Some of the information regarding address information can point to sanction based irregularities previously discussed.

Table 8.3: C104 Parameters

Name	Type	Description	Value
ATTR_NAME	Character List	Attribute Name	"DIRECTOR_RESIDENCE"
MIN_SCORE	Numeric Constant	Minimum Score	0
MAX_SCORE	Numeric Constant	Maximum Score	200
WEIGHT	Numeric Constant	Weight	0.25

```
%cdd_rule_attr_level( attribute_code=&attr_name
, party_type_desc=' ORGANIZATION'
, weight=&weight
, max_score=&max_score
, min_score=&min_score
, auto_high_score=
, rule_score_exp=LOOKUP_SCORE
, not_exceed_range_YN=Y
, auto_high_exp=%nrbrquote(ifc(RULE_SCORE > MAX_SCORE, 'Y', 'N'
))
, triggering_value_var=ATTRIBUTE_CHAR_VALUE
, triggering_value_type_code=CHAR
);
```

8.1.1 Rule Group CDDRG_C020

A sanctions rule group is one that examines for breaches of national and international sanctions on people, goods and countries. Not every sanction rule group contributes the same level of risk to the bank. In our example the developer simplified matters by

creating one group of rules to map to sanctions regulations in EU (2012). However in a situation where others exist the use of minimums and maximum values would be more of a necessity.

8.1.1.1 CDDR_C910

No parameters are utilized here as this manages the maximum value attributed to the overall group. Once all values from each rule within the group are aggregated the score is checked against the maximum value here and if it is exceeded the model uses the score here otherwise the aggregate number. Here a bank would be able to decide and adjust the overall risk attributed to country-based KYC risks and make a determination if it is more/less important to the operations of the bank. Normally a bank would determine this area to be of higher importance when they have more of a multi-geographic presence or if they operate within a region where there is extreme risk of terrorist group activities.

```
/* Country Max Attribute Rule */  
%cdd_rule_attr_max;
```

8.1.2 Rule Group CDDRG_C030

The *country* category group will always be a dynamic category due to the changing nature of sanctions lists. As new lists and risks are identified the logic controlled how those scores impact country risk scores would be considered in this section.

8.1.2.1 CDDR_C920

No Parameters are necessary for this group as it creates the composite score based off of the guiding logic of the maximum and the overall combinatorial logic of the sub-groups. The *PRIMARY_ENTITY_NUMBER* referred to below is indicative of a unique identifier that the bank would use to identify all client activity at the bank through any channel. The code behind this group behaves by doing the actual aggregation of all necessary rules for this specific client.

```

/* Weighted Sum of Country Attribute Scores Divided by
   Weighted Sum of Max Country Score */

/* Weight should be set to 1 if the category level score
   should not be weighted. */
/* A different weight can be applied when calculating the
   overall score. */

%cdd_rule_cat_agg( weight=1
, by_list=PRIMARY_ENTITY_NUMBER
, score_type_code="CAT"
, rule_score_exp=%nrbrquote(ifn(SUM_MAX_SCORE ne 0, SUM_RULE_
  SCORE*WEIGHT / SUM_MAX_SCORE*100, .))
, auto_high_exp=MAX_AUTO_HIGH_IND
);

```

8.1.2.2 CDDR_E101

```

/* Industry Attribute Rule */
%cdd_rule_attr_level( attribute_code=&attr_name
, party_type_desc='ORGANIZATION'
, weight=&weight
, max_score=&max_score
, min_score=&min_score
, auto_high_score=
, rule_score_exp=LOOKUP_SCORE
, not_exceed_range_YN=Y
, auto_high_exp=%nrbrquote(ifc(RULE_SCORE > MAX_SCORE, 'Y', 'N'
  ))
, triggering_value_var=ATTRIBUTE_CHAR_VALUE
, triggering_value_type_code=CHAR
);

```

8.1.2.3 CDDR_E102

Similar to industrial attributes banks also risk rate the details it has with different legal entity types in order to ensure it meets with overall risk goals. As per 8.4, legal entities are a highly utilized mechanism for masking ultimate beneficial owners. Through the practice of layering transactions through shell legal entities a long trail of transactions can successfully clean money and if an institution is not paying attention through vigorous entity resolution mechanisms it can allow transactions to criminal elements. When this occurs regulators are judicious in the fines they impose upon said banks.

Table 8.4: E102 Parameters

Name	Type	Description	Value
ATTR_NAME	Character List	Attribute Name	"LEGAL_ENTITY_TYPE"
WEIGHT	Numeric Constant	Weight	.75
MIN_SCORE	Numeric Constant	Minimum Score	0
MAX_SCORE	Numeric Constant	Maximum Score	100

```

/* Legal Entity Status Rule */
%cdd_rule_attr_level( attribute_code=&attr_name
, party_type_desc='ORGANIZATION'
, weight=&weight
, max_score=&max_score
, min_score=&min_score
, auto_high_score=
, rule_score_exp=LOOKUP_SCORE
, not_exceed_range_YN=Y
, auto_high_exp=%nrquote(ifc(RULE_SCORE > MAX_SCORE, 'Y', 'N'
))
, triggering_value_var=ATTRIBUTE_CHAR_VALUE
, triggering_value_type_code=CHAR
);

```

8.1.2.4 CDDR_E103

One of the saddest aspects of money laundering is the use of charities to mask illicit activities. As charities generally operate around people at risk in war-torn countries they have become a favoured means of terrorist financing as the influx of funding to them can be hard to track and their activities tend to utilize logistic chains to bring goods to high risk geographies. As a result, banks tend to treat non-profit corporations and charities with a risk rating process to validate the actions claimed as humanitarian, 8.5.

Table 8.5: E103 Parameters

Name	Type	Description	Value
ATTR_NAME	Character List	Attribute Name	"NON_PROFIT"
WEIGHT	Numeric Constant	Weight	0.5
MIN_SCORE	Numeric Constant	Minimum Score	0
MAX_SCORE	Numeric Constant	Maximum Score	100

```

/* Non-Profit Organization Attribute Rule */
%cdd_rule_attr_level( attribute_code=&attr_name
, party_type_desc=' ORGANIZATION'
, weight=&weight
, max_score=&max_score
, min_score=&min_score
, auto_high_score=
, rule_score_exp=%nrbrquote( ifc( ATTRIBUTE_CHAR_VALUE=' Y' , 100,
0) )
, not_exceed_range_YN=N
, auto_high_exp=
, triggering_value_var=ATTRIBUTE_CHAR_VALUE
, triggering_value_type_code=CHAR
);

```

8.1.2.5 CDDR_E104

Certain types of financial activity are prone to utilization by money launders. Of particular emphasis are mechanism that can be used that mask identity. As such entities

that have links to money service bureaus or where there is a higher than average usage of certain financial instruments are treated as high risk. The bank measures 'normalcy' for instrument use by assessing a sample of similar businesses and gauging the transaction and product types utilized in their operational profile as per the inputs we see in 8.6.

Table 8.6: E104 Parameters

Name	Type	Description	Value
ATTR_NAME	Character List	Attribute Name	"MONEY_ORDERS" "MSB" "PREPAID_CARDS" "TRAVELERS_CHEQUES" "CURRENCY_EXCHANGE"
WEIGHT	Numeric Constant	Weight	
MAX_SCORE	Numeric Constant	Maximum Score	
MIN_SCORE	Numeric Constant	Minimum Score	

```

/* Money Services Businesses Types Attribute Rule */
%cdd_rule_attr_level( attribute_code=&attr_name
, party_type_desc='ORGANIZATION'
, weight=&weight
, max_score=&max_score
, min_score=&min_score
, auto_high_score=
, rule_score_exp=.
, not_exceed_range_YN=N
, auto_high_exp=%nrbrquote(ifc(ATTRIBUTE_CHAR_VALUE='Y', 'Y', '
  N'))
, triggering_value_var=ATTRIBUTE_CHAR_VALUE
, triggering_value_type_code=CHAR
);

```

8.1.2.6 CDDR_E105

It is difficult to validate identity using the Internet and often gambling entities are more focused on profit than they are on compliance. In a high number of countries money laundering occurs by using both online and physical gaming locations. What occurs is the money launderers create gaming accounts through an influx of prepaid cards or other products where anonymity can be maintained. They conduct low risk gaming for

a nominal amount of time and 'cash out'. The cash out effectively creates *clean* money that can then be used by the launders legitimately. Hence the importance of examining client records where links to gambling are defined as per 8.7.

Table 8.7: E105 Parameters

Name	Type	Description	Value
ATTR_NAME	Character List	Attribute Name	'INTERNET_GAMBLING'
WEIGHT	Character Constant	Weight	1
MAX_SCORE	Numeric Constant	Maximum Score	100
MIN_SCORE	Numeric Constant	Minimum Score	0

```

/* Non-Personal Entity Rule: Engaged in Internet Gambling /
   Casinos Attribute Rule */
%cdd_rule_attr_level( attribute_code=&attr_name
, party_type_desc=' ORGANIZATION'
, weight=&weight
, max_score=&max_score
, min_score=&min_score
, auto_high_score=
, rule_score_exp=.
, not_exceed_range_YN=N
, auto_high_exp=%nrbquote( ifc (ATTRIBUTE_CHAR_VALUE=' Y' , ' Y' , '
   N' ) )
, triggering_value_var=ATTRIBUTE_CHAR_VALUE
, triggering_value_type_code=CHAR
);

```

8.1.2.7 CDDR_E106

Accounts held in trust are another example of a legal instrument that are common in banking that can be used to mask money laundering activities and need to be examined as a key input as per 8.8. Trust accounts can be credited with dividend returns from illicit activity and can be placed in the names of children in order to avoid prosecution. Depending on national legislation and the strength of KYC requirements the trusts may not be obliged to publish all elements of the beneficial owner.

Table 8.8: E106 Parameters

Name	Type	Description	Value
ATTR_NAME	Character List	Attribute Name	'TRUST_ACCOUNT'
WEIGHT	Numeric Constant	Weight	
MAX_SCORE	Numeric Constant	Maximum Score	
MIN_SCORE	Numeric Constant	Minimum Score	

```

/* Trust Account Attribute Rule */
%cdd_rule_attr_level( attribute_code=&attr_name
, party_type_desc='ORGANIZATION'
, weight=&weight
, max_score=&max_score
, min_score=&min_score
, auto_high_score=
, rule_score_exp=.
, not_exceed_range_YN=N
, auto_high_exp=%nrquote(ifc(ATTRIBUTE_CHAR_VALUE='Y', 'Y', '
  N'))
, triggering_value_var=ATTRIBUTE_CHAR_VALUE
, triggering_value_type_code=CHAR
);

```

8.1.2.8 CDDR_E107

Not all governments are democratic and there exists in many countries examples of political despotism, dictatorships or proxy governments. FATF maintains a list of countries that exhibit behaviour patterns that do not put their citizens first and are more totalitarian regimes. Such countries further look to use diplomatic ties to launder money and illegal activities such as meth-amphetamine trafficking as a means of bolstering their exchequers. The activity of consulates and embassies can be party to controlling these activities and warrant due diligence in terms of transactional governance as a result.

```

/* Foreign Consultate Embassy Attribute Rule */
%cdd_rule_attr_level( attribute_code=&attr_name

```

Table 8.9: E107 Parameters

Name	Type	Description	Value
ATTR_NAME	Numeric Constant	Attribute Name	'FOREIGN_CONSULATE_EMBASSY'
WEIGHT	Numeric Constant	Weight	
MAX_SCORE	Numeric Constant	Maximum Score	
MIN_SCORE	Numeric Constant	Minimum Score	

```

, party_type_desc='ORGANIZATION'
, weight=&weight
, max_score=&max_score
, min_score=&min_score
, auto_high_score=
, rule_score_exp=.
, not_exceed_range_YN=N
, auto_high_exp=%nrbrquote(ifc(ATTRIBUTE_CHAR_VALUE='Y', 'Y', '
  N'))
, triggering_value_var=ATTRIBUTE_CHAR_VALUE
, triggering_value_type_code=CHAR
);

```

8.1.2.9 CDDR_E108

Corporations that issue bearer shares are immediately suspect in any credible regulatory environment due to the nature of the financial instrument. Originally these were a common product utilized in trade finance in order to close business internationally prior to modern telecommunications practices. Since the advent of modern telecommunications though the need for instruments of this ilk has declined rapidly. Generally, these are used to mask beneficial ownerships as they are used as cash equivalents and needed to be examined. Hence the inclusion below in 8.10

Table 8.10: E108 Parameters

Name	Type	Description	Value
ATTR_NAME	Character List	Attribute Name	'ISSUES_BEARER_SHARES'
WEIGHT	Numeric Constant	Weight	
MAX_SCORE	Numeric Constant	Maximum Score	
MIN_SCORE	Numeric Constant	Minimum Score	

```

/* Non-Personal Entity Rule: Corporations that issue Bearer
   Shares Attribute Rule */
%cdd_rule_attr_level( attribute_code=&attr_name
, party_type_desc='ORGANIZATION'
, weight=&weight
, max_score=&max_score
, min_score=&min_score
, auto_high_score=
, rule_score_exp=.
, not_exceed_range_YN=N
, auto_high_exp=%nrquote(ifc(ATTRIBUTE_CHAR_VALUE='Y', 'Y', '
  N'))
, triggering_value_var=ATTRIBUTE_CHAR_VALUE
, triggering_value_type_code=CHAR
);

```

8.1.2.10 CDDR_E109

A *PEP* or 'Politically Exposed Person' is a term given to individuals legal entities with links to individuals in roles that are deemed to be high risk (8.11). Classical examples are government leaders, parliamentarians and bureaucrats of all descriptions. At a corporate level these people become interesting in terms of the corrupt practices act where influence pedlars bribe these individuals. To hide the trail of illicit influence the politicians create shell corporations often in a family members name and establish said family member as a director in order to hide the proceeds of suspicious activity. This mechanism is also used for the purposes of off shoring income to avoid taxation regulation. In comparison to **CDDR_E202** this section examines a legal entity whereas **CDDR_E202** looks at individuals.

```

/* Non-Personal Entity Rule: Politically Exposed Person (PEP)
   - Foreign Domestic */
%cdd_rule_attr_level( attribute_code=&attr_name
, party_type_desc='ORGANIZATION'
, weight=&weight
, max_score=&max_score

```

Table 8.11: E109 Parameters

Name	Type	Description	Value
ATTR_NAME	Character List	Attribute Name	'PEP'
WEIGHT	Numeric Constant	Weight	
MAX_SCORE	Numeric Constant	Maximum Score	
MIN_SCORE	Numeric Constant	Minimum Score	

```

, min_score=&min_score
, auto_high_score=
, rule_score_exp=%nrbrquote(ifc(ATTRIBUTE_CHAR_VALUE='Y',.,0))
, not_exceed_range_YN=N
, auto_high_exp=%nrbrquote(ifc(ATTRIBUTE_CHAR_VALUE='Y', 'Y', '
  N'))
, triggering_value_var=ATTRIBUTE_CHAR_VALUE
, triggering_value_type_code=CHAR
);

```

8.1.2.11 CDDR_E110

Negative news in isolation is not a credible means of isolating money laundering activity. However, in terms of client risk rating negative news can be used to highlight illicit activities that a client may have obfuscated from a financial institution regarding source of wealth. This area has become a focus area for financial institutions in order to data-mine for potential risks within their client base. The signal to noise ratio associated with negative news means that each artifact hit that occurs has to be validated thoroughly as the false positive rate is high. 8.12 looks at negative news references associated with a specific client record.

Table 8.12: E110 Parameters

Name	Type	Description	Value
ATTR_NAME	Character List	Attribute Name	'NEGATIVE_NEWS'
WEIGHT	Numeric Constant	Weight	
MAX_SCORE	Numeric Constant	Maximum Score	
MIN_SCORE	Numeric Constant	Minimum Score	

```

/* Non-Personal Entity Rule: Negative Media Search Rule */
%cdd_rule_attr_level( attribute_code=&attr_name
, party_type_desc='ORGANIZATION'
, weight=&weight
, max_score=&max_score
, min_score=&min_score
, auto_high_score=
, rule_score_exp=.
, not_exceed_range_YN=N
, auto_high_exp=%nrquote(ifc(ATTRIBUTE_CHAR_VALUE='Y', 'Y', '
  N'))
, triggering_value_var=ATTRIBUTE_CHAR_VALUE
, triggering_value_type_code=CHAR
);

```

8.1.2.12 CDDR_E201

This grouping, 8.13, looks at the individual rather than a legal entity. In the same way as industry ties have certain risks associated with them the same is true for occupations. As such banks often create a correlation between client occupation codes and risk tolerances. Occupation codes allow the bank to segment their population and establish a mean income for the segment. When occurrences for a client are beyond that established segment mean it becomes a highlight for enhanced due diligence requirements.

Table 8.13: E201 Parameters

Name	Type	Description	Value
WEIGHT	Numeric Constant	Weight	1
ATTR_NAME	Character List	Attribute Name	"OCCUPATION"
MIN_SCORE	Numeric Constant	Minimum Score	0
MAX_SCORE	Numeric Constant	Maximum Score	100

```

/* Occupation Attribute Rule */
%cdd_rule_attr_level( attribute_code=&attr_name
, party_type_desc='INDIVIDUAL'

```

```

, weight=&weight
, max_score=&max_score
, min_score=&min_score
, auto_high_score=
, rule_score_exp=LOOKUP_SCORE
, not_exceed_range_YN=Y
, auto_high_exp=%nrquote(ifc(RULE_SCORE > MAX_SCORE, 'Y', 'N'
))
, triggering_value_var=ATTRIBUTE_CHAR_VALUE
, triggering_value_type_code=CHAR
);

```

8.1.2.13 CDDR_E202

As per the entity group this ties the individual to political exposure. Further, it links the client population as relatives or close associates to known political entities/persons. Family members often become a trusted means to launder ill gotten gains and mask financial activity or corruption, thus it is important that banks monitor their client populations for changes in status for PEPs through using inputs like 8.14.

Table 8.14: E202 Parameters

Name	Type	Description	Value
ATTR_NAME	Character List	Attribute Name	"PEP"
MAX_SCORE	Numeric Constant	Maximum Score	
MIN_SCORE	Numeric Constant	Minimum Score	
WEIGHT	Numeric Constant	Weight	

```

/* Personal Entity Rule: PEP Attribute Rule */
%cdd_rule_attr_level( attribute_code=&attr_name
, party_type_desc='INDIVIDUAL'
, weight=&weight
, max_score=&max_score
, min_score=&min_score
, auto_high_score=
, rule_score_exp=.
, not_exceed_range_YN=N

```

```

, auto_high_exp=%nrquote(ifc(ATTRIBUTE_CHAR_VALUE='Y', 'Y', '
  N'))
, triggering_value_var=ATTRIBUTE_CHAR_VALUE
, triggering_value_type_code=CHAR
);

```

8.1.2.14 CDDR_E203

The *"Personal Entity Rule: Negative Media News Search Attribute Rule"* acts in the same way as the legal entity based negative news indicator but acts on the individual. This indicator is even more dangerous in terms of false positive generation as name commonality allows for unwarranted escalations to occur with ease. Hence the need for a robust entity resolution program to establish identity of the client versus the reported individual in the negative news excerpt. 8.15 assumes that such entity resolution has occurred prior to examination of the client record.

Table 8.15: E203 Parameters

Name	Type	Description	Value
ATTR_NAME	Character List	Attribute Name	'NEGATIVE_NEWS'
WEIGHT	Numeric Constant	Weight	
MAX_SCORE	Numeric Constant	Maximum Score	
MIN_SCORE	Numeric Constant	Minimum Score	

```

/* Personal Entity Rule: Negative Media News Search Attribute
  Rule */
%cdd_rule_attr_level( attribute_code=&attr_name
, party_type_desc='INDIVIDUAL'
, weight=&weight
, max_score=&max_score
, min_score=&min_score
, auto_high_score=
, rule_score_exp=.
, not_exceed_range_YN=N
, auto_high_exp=%nrquote(ifc(ATTRIBUTE_CHAR_VALUE='Y', 'Y', '
  N'))

```

```

, triggering_value_var=ATTRIBUTE_CHAR_VALUE
, triggering_value_type_code=CHAR
);

```

8.1.2.15 CDDR_E204

A *SAR* (Suspicious Activity Report) or *STR* (Suspicious Transaction Report) is a regulatory filing that a bank is obligated to send to regulators when activity within the transactional feeds of an individual warrants investigation. Generally this is triggered during conventional transaction monitoring scoring when a scenario trigger (e.g. personal cash, velocity, large cash transaction) occurs. These trigger case investigations that have a final state of either closure or the submission of a regulatory filing. When an individual has transactions that are under review and have filings then it makes sense for the the bank to maintain a corollary risk rating at the client level. 8.16 looks at the counts of historical SARs associated with a client record.

Table 8.16: E204 Parameters

Name	Type	Description	Value
ATTR_NAME	Character Constant	Attribute Name	'SAR_COUNT'
WEIGHT	Numeric Constant	Weight	
MAX_SCORE	Numeric Constant	Maximum Score	
MIN_SCORE	Numeric Constant	Minimum Score	

```

/* Entity Rule: Number of SARs filed */
%cdd_rule_attr_level( attribute_code=&attr_name
, party_type_desc=
, weight=&weight
, max_score=&max_score
, min_score=&min_score
, auto_high_score=
, rule_score_exp=.
, not_exceed_range_YN=N
, auto_high_exp=%nrbquote(ifc(input(ATTRIBUTE_CHAR_VALUE,
BEST32.) > 0, 'Y', 'N'))

```

```
, triggering_value_var=%nrquote(input(ATTRIBUTE_CHAR_VALUE,  
    BEST32.))  
, triggering_value_type_code=NUM  
);
```

8.1.3 Rule Group CDDRG_E020

This group looks at the logic around setting a minimum or maximum value for the associated entity based category.

8.1.3.1 CDDR_E910

No parameters are required but similar to the previous grouping around country indicators this rule group is used to set a maximum score for entity type risks. This means that the maximum score is compared against the aggregate score generated and the model determines which one is used based on the values presented.

```
%cdd_rule_attr_max;
```

8.1.4 Rule Group CDDRG_E030

We examine all combinatorial logic associated with the construction of a category score.

8.1.4.1 CDDR_E920

This group creates a composite score for the entity risk categories discussed and rates it against the logical surrounding the category maximum in the previous group using the logic determined by the bank and its risk appetite.

```
/* Weight should be set to 1 if the category level score  
    should not be weighted. */  
/* A different weight can be applied when calculating the  
    overall score. */  
  
%cdd_rule_cat_agg( weight=1
```

```

,by_list=PRIMARY_ENTITY_NUMBER
,score_type_code="CAT"
, rule_score_exp=%nrbrquote(ifn(SUM_MAX_SCORE ne 0, SUM_RULE_
    SCORE*WEIGHT / SUM_MAX_SCORE*100, .))
, auto_high_exp=MAX_AUTO_HIGH_IND
);

```

8.1.4.2 CDDR_P010

```

/* Weight should be set to 1 if the category level score
   should not be weighted. */
/* A different weight can be applied when calculating the
   overall score. */

%cdd_rule_cat_agg( weight=1
,by_list=PRIMARY_ENTITY_NUMBER
,score_type_code="CAT"
, rule_score_exp=%nrbrquote(ifn(SUM_MAX_SCORE ne 0, SUM_RULE_
    SCORE*WEIGHT / SUM_MAX_SCORE*100, .))
, auto_high_exp=MAX_AUTO_HIGH_IND
);

```

8.1.5 Rule Group CDDRG_P020

The *product* category group should also employ mechanisms examining variables such that floor and ceiling values can be applied with ease.

8.1.5.1 CDDR_P910

As with previous categories this rule group looks at setting a category maximum risk score for the all groupings with the product domain.

```

%cdd_rule_attr_max;

```

8.1.6 Rule Group CDDRG_P030

As the bank updates the groupings further product sub-groupings could be instantiated due to legislative changes. This section examines combining all sub-groups into score elements.

8.1.6.1 CDDR_P920

This is the composite score creation mechanism we've used previously in order to establish a category score for the group. In this case the bias is around product utilization and the banks offered product range and how it risk scores them.

```
/* Weight should be set to 1 if the category level score
   should not be weighted. */
/* A different weight can be applied when calculating the
   overall score. */

%cdd_rule_cat_agg(weight=1
,by_list=PRIMARY_ENTITY_NUMBER
,score_type_code="CAT"
, rule_score_exp=%nrquote(ifn(SUM_MAX_SCORE ne 0, SUM_RULE_
   SCORE*WEIGHT / SUM_MAX_SCORE*100, .))
, auto_high_exp=MAX_AUTO_HIGH_IND
);
```

8.1.6.2 CDDR_X010

```
/* Calculate Sum of Weighted Category Scores */
length TEMP_RULE_SCORE 8.;
retain TEMP_RULE_SCORE 8.;
drop TEMP_RULE_SCORE;

length MAX_AUTO_HIGH_IND $ 1;
retain MAX_AUTO_HIGH_IND;
drop MAX_AUTO_HIGH_IND;

if first.PRIMARY_ENTITY_NUMBER then do;
```

```

TEMP_RULE_SCORE = 0;
MAX_AUTO_HIGH_IND='';
end;

if AUTO_HIGH_IND > MAX_AUTO_HIGH_IND then MAX_AUTO_HIGH_IND=
  AUTO_HIGH_IND;
if CATEGORY_CODE = "COUNTRY" then TEMP_RULE_SCORE = TEMP_RULE_
  SCORE + RULE_SCORE * &country_weight;
if CATEGORY_CODE = "ENTITY" then TEMP_RULE_SCORE = TEMP_RULE_
  SCORE + RULE_SCORE * &entity_weight;
if CATEGORY_CODE = "PRODUCT" then TEMP_RULE_SCORE = TEMP_RULE_
  SCORE + RULE_SCORE * &product_weight;

if last.PRIMARY_ENTITY_NUMBER then do;
CATEGORY_CODE='';
RULE_SCORE = ROUND(TEMP_RULE_SCORE,1);
AUTO_HIGH_IND=MAX_AUTO_HIGH_IND;
output;
end;

```

8.1.6.3 CDDR_X015

```

/* Apply weight based on customer age */
if RULE_SCORE ne . then do;
if CUSTOMER_SINCE_DATE eq .
then do;
/*use missing date weight*/
RULE_SCORE = ROUND(RULE_SCORE * &missing_age_weight,1);
end;
else do;
if yrdif(datepart(CUSTOMER_SINCE_DATE), today(), 'AGE') <= &
  new_customer_age
then do;
RULE_SCORE = ROUND(RULE_SCORE * &new_customer_weight,1);
end;
end;

```

```
end;
```

8.1.6.4 CDDR_X020

```
/* Assign Score Classification */
if RULE_SCORE <= &l_max then SCORE_CLASSIFICATION_CODE = 'L';
else if RULE_SCORE <= &m_max then SCORE_CLASSIFICATION_CODE =
  'M';
else SCORE_CLASSIFICATION_CODE = 'H';

/* Auto High flag overrides score classification */
if AUTO_HIGH_IND eq 'Y' then SCORE_CLASSIFICATION_CODE = 'H';

/* Use the override rating if the new score is close to the
   last review score */
if CDD_REVIEW_FLG ne '1' and
SCORE_CLASSIFICATION_CODE='H' and
OVERRIDE_SCORE_CLASS_CODE ne SCORE_CLASSIFICATION_CODE and
OVERRIDE_SCORE_CLASS_CODE ne '' and
( RULE_SCORE <= LAST_REVIEW_SCORE or
(RULE_SCORE - LAST_REVIEW_SCORE) <= &score_diff
)
then do;
SCORE_CLASSIFICATION_CODE = OVERRIDE_SCORE_CLASS_CODE;
end;
```

8.1.6.5 CDDR_X030

```
/* Suppression Rule */

if FINAL_DISPOSITION_CODE='RETAIN' and
CDD_REVIEW_FLG ne '1' and
SCORE_CLASSIFICATION_CODE='H' and
( RULE_SCORE <= LAST_REVIEW_SCORE or
(RULE_SCORE - LAST_REVIEW_SCORE) <= &score_diff
```

```
) then do;  
CDD_SUPPRESSION_FLG='1';  
end;  
else do;  
CDD_SUPPRESSION_FLG='0';  
end;
```

9

Appendix 2

9.1 A Survey on understanding software specifications written in formal methods

9.1.1 Instructions

The following survey is aimed at financial crime software professionals. It aims to understand if formal methods, a means of describing software using mathematical expressions, could provide lift if financial crime software was described using them. We examine one specific type of formal method in our study, the Trace Function Method (Parnas and Dragomiroiu, 2006), but are interested to understand if formal methods in any form are in use in your organizations.

As you read through this survey please keep in mind a few basic assumptions. We are trying to understand if the mathematics are easily read and comprehended or if the opposite is in fact true. We are interested also to learn if you see potential wins for your business if these methods were put into practice in your industry or if they would hinder your ability to sell. We assume that you have an understanding of the mathematics included but would be happy to answer any questions you have and clarify.

9.1.2 Introduction

We utilize some example material and excerpts from the following paper just to set expectations regarding the approach: Parnas (2006)

Is it possible to have financial crime specifications that are both precise and human readable? Precision generally comes at a price and that is the associated readability of the specification. For example the following is precise but is quite difficult for a human being to parse;

$$\boxed{((\exists i, B[i] = 'x') \wedge (B[j'] = x) \wedge (\text{present}' = \text{true})) \vee ((\forall i, ((1 \leq i \leq N) \Rightarrow B[i] \neq x)) \wedge (\text{present}' = \text{false})) \wedge ('x = x' \wedge 'B = B')}$$

Frankly speaking, unless it was due to stringent requirements few in industry would consider the above example a statement that is easy to parse and comprehend and this is just an example. However in the world of regulatory compliance precision and readability are vitally important to showcase accuracy in meeting regulatory requirements. The following example is easily readable but lacks precision;

"Set i to indicate the place in the array B where x can be found and set present to be true. Otherwise set present to be false"

It introduces vague statements that would leave a developer unclear on how to proceed. Vague statements within the financial crime domain would open a financial institution to compliance related regulatory risk which could result in severe penalties. Thus we arrive at a format that has the potential to be both precise and readable;

	$(\exists i, B[i] = x)^a$	$(\forall i, \neg(B[i] = x))$	
j'	$B[j'] = x$	<i>true</i>	$\wedge NC(x, B)$
$\text{present}' =$	true	false	

a. These tables depend on using a logic in which evaluating a partial function outside of its domain yields "*false*" for all built-in predicates.

The above example is both easy to parse whilst remaining mathematically correct. The goal of this survey is to study the validity of this statement.

9.1.3 Date Management Software Example

The following example looks at a simple program for managing dates. It has a series of simple inputs and output variables. As you can imagine it will need several functions to save and get days, months and years. It also will require some utility functions.

Date Module

Output Variables

Variable Name	Type
<id>.day	<integer>
<id>.month	<integer>
<id>.year	<integer>
<id>. Value	<integer>

Input Variables

Variable Name	Type
PGM	<program name>
in1	<integer>
in2	date

Access Programs

Program Name	Oname	Value	in1	in2	Abbreviated Event Descriptor
SETDAY	<id>		<integer>		(PGM.SETDAY, 'in, day')
SETPMONTH	<id>		<integer>		(PGM.SETMONTH, 'in, month',)
SETYEAR	<id>		<integer>		(PGM.SETYEAR, 'in, year')
GETDAY	<id>	<integer>			(PGM.GETDAY, Value, 'day')
GETMONTH	<id>	<integer>			(PGM.GETMONTH, Value, 'month')
GETYEAR	<id>	<integer>			(PGM.GETYEAR, Value, 'year')
NEWDATE	<id>			<id>	(PGM.NEWDATE, '<in2>, <in2>')
DELETEDATE			<id>		(PGM.DELETEDATE, '<in2>, <in2>')
COPYDATE			<id>		(PGM.COPYDATE, '<in2>')

Auxiliary Functions

day(T) =

$\neg(T = _) \wedge$	(T = $_$) \vee PGM(σ (T) = NEWDATE)	0
	(PGM(σ (T)) = SETDAY)	'in(σ (T))
	(PGM(σ (T)) = COPYDATE)	day('in2(σ (T))
	(PGM(σ (T)) = DELETEDATE)	
	\neg (PGM(σ (T)) = SETDAY \vee PGM(σ (T)) = COPYDATE \vee PGM(σ (T)) = DELETEDATE \vee PGM(σ (T)) = NEWDATE)	day(p(T))

month(T) =

$\neg(T = _) \wedge$	(T = $_$) \vee PGM(σ (T) = NEWDATE)	0
	(PGM(σ (T)) = SETMONTH)	'in(σ (T))
	(PGM(σ (T)) = COPYDATE)	month('in2(σ (T))
	(PGM(σ (T)) = DELETEDATE)	
	\neg (PGM(σ (T)) = SETMONTH \vee PGM(σ (T)) = COPYDATE \vee PGM(σ (T)) = DELETEDATE \vee PGM(σ (T)) = NEWDATE)	month(p(T))

year(T) =

$\neg(T = _) \wedge$	(T = $_$) \vee PGM(σ (T) = NEWDATE)	0
	(PGM(σ (T)) = SETYEAR)	'in(σ (T))
	(PGM(σ (T)) = COPYDATE)	year('in2(σ (T))
	(PGM(σ (T)) = DELETEDATE)	
	\neg (PGM(σ (T)) = SETYEAR \vee PGM(σ (T)) = COPYDATE \vee PGM(σ (T)) = DELETEDATE \vee PGM(σ (T)) = NEWDATE)	year(p(T))

Value (T) ≡

PGM(r(T)) = GETDAY	day'(T)
PGM(r(T)) = GETYEAR	year'(T)
PGM(r(T)) = GETMONTH	month'(T)
$\neg (\text{PGM}(r(T)) = \text{GETDAY} \vee \text{PGM}(r(T)) = \text{GETYEAR} \vee \text{PGM}(r(T)) = \text{GETMONTH})$	

Output Functions

<id>.day ≡ day(T<id>)

<id>.month ≡ month(T<id>)

<id>.year ≡ year(T<id>)

<id>.Value ≡ Value(T<id>)

9.1.4 Time Management Software Example

The following example is similar to the previous one but focuses instead on time storage rather than dates. Time storage means hours and minutes are stored in this case example, like a simple watch mechanism.

Time storage module

Output Variables

Variable Name	Type
hr	<integer>
min	<integer>

Access Programs

Program Name	'in	Abbreviated Event Descriptor
SET HR	<integer>	(PGM:SET HR, 'in, hr')
SET MIN	<integer>	(PGM:SET MIN, 'in, min')
INC		(PGM:INC, hr', min')
DEC		(PGM:DEC, hr', min')

Output Functions

hr(T) ≡		
PGM _(T) = SET HR ∧	0 ≤ 'in(r(T)) < 24	'in(r(T))
	¬ (0 ≤ 'in(r(T)) < 24)	hr((p(T)))
PGM _(T) = SET MIN		hr((p(T)))

PGM _(T) = INC ∧	min(p(T)) = 59 ∧	hr(p(T)) = 23	0
		¬ hr(p(T)) = 23	1 + hr((p(T)))
	¬ (min(p(T)) = 59)		
PGM _(T) = DEC ∧	¬ (min(p(T)) = 0)		hr((p(T)))
	min(p(T)) = 0 ∧	¬ (hr(p(T))) = 0	hr((p(T))) - 1
		hr(p(T)) = 0	23
T = _			0

min(T) ≡

PGM _(T) = SET HR		min(p(T))
PGM _(T) = SET MIN ∧	0 ≤ 'in(r(T)) ≤ 59	'in(r(T))
	¬ (0 ≤ 'in(r(T)) ≤ 59)	min(p(T))
PGM _(T) = INC ∧	min(p(T)) = 59	0
	¬ (min(p(T)) = 59)	min(p(T)) + 1
PGM _(T) = DEC ∧	¬ (min(p(T)) = 0)	min((p(T))) - 1
	min(p(T)) = 0	59
T = _		0

9.1.5 Stack Example

The last example is a simple stack from basic programming concepts. It provides the traditional functions, inputs and outputs.

A stack of limited range and depth

Output Variables

Variable Name	Type
top	<integer>
depth	<integer>
exc	(none, range, depth, empty)
Value	<integer>

Access Programs

Program Name	'Value	'in	Abbreviated Event Descriptor
PUSH		<integer>	(PGM.PUSH, 'in, top', depth', exc)
POP			(PGM.POP, top', depth', exc)
TOP	<integer>		(PGM.TOP, Value', exc)
DEPTH	<integer>		(PGM.DEPH, Value')

Auxiliary Functions

inrange(i) = LB ≤ i ≤ UB
 noeffect(e) = (PGM(e) = PUSH ∧ ¬inrange('in(e))) ∨ PGM(e) = TOP ∨ PGM(e) = DEPTH
 full(T) = depth(T) = d
 empty(T) = depth(T) = 0

ps(T1, T2) =

T2 = _			T1	
(T2 ≠ _) ∧ noeffect(o(T2))			ps(T1, s(T2))	
(T2 ≠ _) ∧ noeffect(o(T2)) ∧	PGM(o(T2)) = PUSH ∧	full(T1)	ps(T1, s(T2))	
		¬full(T1)	ps(T1, o(T2), s(T2))	
	PGM(o(T2)) = POP ∧	¬empty(T1)	ps(p(T1), s(T2))	
		empty(T1)	ps(T1, s(T2))	

strip(T) = ps(_, T)

Output variable functions

top(T) =

strip(T) = _	
strip(T) ≠ _	'in(r(strip(T)))

Value(T) =

PGM(r(T)) = TOP	top(p(T))
PGM(r(T)) = DEPTH	depth(p(T))
PGM(r(T)) = PUSH	
PGM(r(T)) = POP	

depth(T) =

T = _			0
noeffect(r(T))			depth(p(T))
(T ≠ _) ∧ ¬noeffect(r(T)) ∧	PGM(r(T)) = POP ∧	depth(p(T)) = 0	0
		depth(p(T)) ≠ 0	depth(p(T)) - 1
	PGM(r(T)) = PUSH ∧	depth(p(T)) = d	d
	depth(p(T)) ≠ d	depth(p(T)) + 1	

exc(T) =

PGM(r(T)) = PUSH ∧	¬inrange('in(r(T)))		range
	inrange('in(r(T))) ∧	L(strip(p(T))) = d	depth
		¬(L(strip(p(T))) = d)	none
(PGM(r(T)) = POP ∨ PGM(r(T)) = TOP) ∧	L(strip(p(T))) = 0		empty
	¬(L(strip(p(T))) = 0)		none
PGM(r(T)) = DEPTH			none

9.1.6 A CDD Program Scoring Banking Products

Consider the following example from a CDD Model scoring products for a bank. We examine a number of component elements from a example written using SAS code and being executed in a modeling harness.

9.1.6.1 CDDR_P010

This rule examines all the products that a bank offers and rates it against a risk ranked scale of how the bank assesses them. The bank changes this hot-list often as new products are created/retired and as a result is often in flux. Given the changing nature of banking, moving from a physical interactions to digital, banks are increasingly wary of ensuring more security controls for product offerings in order to meet regulatory needs. This however counteracts customer experience measures and profitability as it inhibits a clients ease of use for new banking products.

Table 9.1: P010 Parameters

Name	Type	Description	Value
ATTR_NAME	Character Constant	Attribute Name	"PRODUCT"
MAX_SCORE	Numeric Constant	Maximum Score	200
MIN_SCORE	Numeric Constant	Minimum Score	0
WEIGHT	Numeric Constant	Weight	1

9.1.6.2 CDDR_P010

```

/* Weight should be set to 1 if the category level score
   should not be weighted. */
/* A different weight can be applied when calculating the
   overall score. */

%cdd_rule_cat_agg( weight=1
,by_list=PRIMARY_ENTITY_NUMBER
,score_type_code="CAT"
, rule_score_exp=%nrquote(ifn(SUM_MAX_SCORE ne 0, SUM_RULE_
SCORE*WEIGHT / SUM_MAX_SCORE*100, .))

```

```
, auto_high_exp=MAX_AUTO_HIGH_IND
);
```

9.1.6.3 CDDR_P910

```
%cdd_rule_attr_max;
```

9.1.6.4 CDDR_P920

```
/* Weight should be set to 1 if the category level score
   should not be weighted. */
/* A different weight can be applied when calculating the
   overall score. */

%cdd_rule_cat_agg(weight=1
,by_list=PRIMARY_ENTITY_NUMBER
,score_type_code="CAT"
, rule_score_exp=%nrquote(ifn(SUM_MAX_SCORE ne 0, SUM_RULE_
  SCORE*WEIGHT / SUM_MAX_SCORE*100, .))
, auto_high_exp=MAX_AUTO_HIGH_IND
);
```

The above code is an example from a CDD model that is evaluating risk scores associated with products. Here is an alternative perspective of the same specified code presented in a formal representation that provides the program inputs, set elements, functions and some traces regarding the functions to explain their operation.

9.2, is a program focused on scoring banking products as part of an overall CDD program. The domain in this case examines the product mix that a client may be taking advantage of from the banking institution i.e. lines of credit, corporate loans, commercial payments, correspondent banking, wealth management. These products have different risk ratings and this program allows the banks analysts to adjust them accordingly.

Table 9.2: PGM(ProductScore)

Name	Type	Range
ATTR_NAME	< enum >	
MAX_SCORE	< int>	
MIN_SCORE	< int >	
WEIGHT	< int >	

The set of elements, 9.3, provides the set of data assets necessary for the program component discussed earlier, 9.2. Product elements are a much smaller asset tailored to each bank and their marketplace and clientele.

Table 9.3: Elements of Set ProductScore

Element Name
PRODUCT
MAX_AUTO_HIGH_IND
CAT
LOOKUP_SCORE

9.4 provides a listing that is centric to auxiliary functions that are necessary for the product category of the CDD model to function as designed.

Table 9.4: ProductScore Auxiliary Functions

i	ProductScore(i)
1	CDDRG_(P010)
2	CDDRG_(P020)
3	CDDRG_(P030)

Profile Questions

Are you?

- A developer
- A product manager
- A domain expert/Subject Matter Expert

How many full-time software developers does your company employ?

Briefly describe the type of financial crime software your company develops.

Does your company use formal methods in any aspect of its software development?

- Yes
- No
- Unaware if we do.

If the answer is no, please indicate why. (Select all that apply.)

- Never considered using formal methods.
- Considered using formal methods, but determined use would not be cost effective.
- Considered using formal methods, but determined that employees do not have necessary skills.
- Other Reasons? Please state.

Overall, indicate if you feel that the use of formal methods would be beneficial in your company. By beneficial we mean that the use of formal methods could lead to improvements in the cost, development time, and/or quality of software, compared to software developed without using formal methods.

- Not beneficial
- Somewhat beneficial
- Moderately beneficial
- Very beneficial

Do you think that the tools and techniques used for different development activities in your organization would complement the use of formal method based descriptions?

9.1.7 Comprehension Questions

Overall, how easy to read were the examples provided above?

- Very Easy
- Easy
- Fair
- Difficult
- Very Difficult

How many of the examples were you able to read through and understand?

- None
- 1
- 2
- All

Would you describe the examples as precise in their descriptions?

- No
- Somewhat
- Yes

Why?

Given the timescale and size of a project undertaken for financial crime software development do you think more precise description would provide lift? Why?

9.1.8 Regulatory Questions

Would providing the experience and training provided to the personal involved in development be beneficial if it results in a marked improvement in software quality and regulatory acceptance of your product? Why?

How complex is it currently to meet regulatory requirements using software?

- Easy: Does add considerable overhead to how we design our software
- Average: Does add some overhead but we manage it appropriately.
- Complex: Does add considerable overhead and we have difficulty managing it appropriately.

Why is that?

How important is precise software to your client base?

- Very important
- Somewhat Important
- Not Important

Do your clients current understand the specifics of your software and how meets regulatory requirements?

- No
- Somewhat
- Yes

Do your clients question the accuracy of compliance aspects of your software?

- Often
- Rarely
- Frequently

Why?

Do your regulators understand how your software functions and agree it meets regulations?

- Yes
- No
- Somewhat

How do you currently education your clients/regulators regarding the specifics of your software solutions?

9.1.9 Utility Questions

Would you consider the formal representations of CDD software useful? Why?

Do you consider them comprehensible?

Did the CDD Product example reduce or increase the complexity to understand the program?

Any further remarks?

Given our previous questions and introduction to this form of the accuracy of these representations and mathematical verification options available to ensure accuracy of a specification do you see potential for using the latter form of representation in financial crime software specifications. Please Discuss.

Bibliography

- Abrial, J. (1996), *The B-book: Assigning Programs to Meanings*, Cambridge University Press, New York, NY, USA. 12, 32, 157
- Aghion, P. and Howitt, P. (1990), ‘A model of growth through creative destruction’. 45
- Angelopoulou, O., Self, R. J. and Flores, D. A. (2012), ‘Combining digital forensic practices and database analysis as an anti-money laundering strategy for financial institutions’, *2013 Fourth International Conference on Emerging Intelligent Data and Web Technologies* **00**, 218–224. 40
- Apache (2018), ‘<http://lucene.apache.org/>’, Website. 72
- Archer, M. (2000), ‘TAME: using PVS strategies for special-purpose theorem proving’, **29**(1-4), 139–181. 29, 30
- Badali, G. A. (2010), ‘Filthy lucre: confronting the risks of money laundering’, *Risk Management* **57**(4), 34. 37
- Bane, D. (2008), Design and documentation of the kernel of a set of tools for working with tabular mathematical expressions, Master’s thesis, University of Limerick. 28, 46, 80, 105
- Bane, D., Jin, Y. and Parnas, D. (2004), Mathematical model of tabular expressions, Technical Report SQRL, Software Quality Research Laboratory. 13, 15, 19, 20, 22, 33, 81, 82, 157, 159
- Barnes, J. E. (2011), ‘Experiences in the industrial use of formal methods’, *Electronic Communications of the EASST* **46**. 145, 147
- Barr, A. J. and Goodnight, J. H. (1976), *A User’s Guide to SAS*, SAS. 72, 75, 85, 95
- Barrett, C. and Tinelli, C. (2007), CVC3, in W. Damm and H. Hermanns, eds, ‘Proceedings of the 19th International Conference on Computer Aided Verification (CAV ’07)’, Vol. 4590 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 298–302. Berlin, Germany. 28

- Barse, E., Kvarnstrom, H. and Jonsson, E. (2003), Synthesizing test data for fraud detection systems, *in* 'Computer Security Applications Conference, 2003. Proceedings. 19th Annual', pp. 384–394. 66
- Bartussek, W. and Parnas, D. L. (1978), *Using assertions about traces to write abstract specifications for software modules*, Springer Berlin Heidelberg, pp. 211–236. 10, 12, 32
- Batista, G. E. A. P. A., Prati, R. C. and Monard, M. C. (2005), Balancing strategies and class overlapping, *in* 'Proceedings of the 6th International Conference on Advances in Intelligent Data Analysis', IDA'05, Springer-Verlag, Berlin, Heidelberg, pp. 24–35. 70
- Behdad, M., Barone, L., French, T. and Bennamoun, M. (2010), An investigation of real-valued accuracy-based learning classifier systems for electronic fraud detection, *in* 'Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation', GECCO '10, ACM, New York, NY, USA, pp. 1893–1900. 44
- Bharadwaj, R. and Sims, S. (2000), Salsa: Combining constraint solvers with bdds for automatic invariant checking, *in* 'Tools and Algorithms for Construction and Analysis of Systems, 6th International Conference, TACAS 2000, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS 2000, Berlin, Germany, March 25 - April 2, 2000, Proceedings', pp. 378–394. 29
- Black, P. E. (2005), 'Greedy algorithm', *Dictionary of Algorithms and Data Structures* 2. 62
- Bolton, R. J. and Hand, D. J. (2001), Unsupervised profiling methods for fraud detection, *in* 'Proc. Credit Scoring and Credit Control VII', pp. 5–7. 5, 48, 50, 52, 62
- Bolton, R. J. and Hand, D. J. (2002), 'Statistical fraud detection: A review'. 5, 49, 50, 51, 52, 55

- Borsuk, M. E., Stow, C. A. and Reckhow, K. H. (2004), 'A bayesian network of eutrophication models for synthesis, prediction, and uncertainty analysis', **173**(2-3), 219–239.
URL: <http://www.sciencedirect.com/science/article/pii/S0304380003003958> 68, 79
- Bowen, J. (1992), *Select Z Bibliography*, Springer London, pp. 367–397. 11
- Bowyer, K. W., Chawla, N. V., Hall, L. O. and Kegelmeyer, W. P. (2011), 'SMOTE: synthetic minority over-sampling technique', **abs/1106.1813**.
URL: <http://arxiv.org/abs/1106.1813> 70
- Brause, R., Langsdorf, T. and Hepp, M. (1999), Neural data mining for credit card fraud detection, in 'Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence', ICTAI '99, IEEE Computer Society, pp. 103–.
URL: <http://dl.acm.org/citation.cfm?id=850950.853684> 65
- Breiman, L. (1996), 'Heuristics of instability and stabilization in model selection', **24**(6), 2350–2383. 71
- Breunig, M. M., Kriegel, H.-P., Ng, R. T. and Sander, J. (2000), 'Lof: Identifying density-based local outliers', *SIGMOD Rec.* **29**(2), 93–104.
URL: <http://doi.acm.org/10.1145/335191.335388> 54
- Cai, X. and Lyu, M. R. (2005), 'The effect of code coverage on fault detection under different testing profiles', *ACM SIGSOFT Software Engineering Notes* **30**(4), 1–7. 141
- Cheng, J. and Jones, C. (1990), On the usability of logics which handle partial functions, Technical Report Series UMCS 90-3-1, Department of Computer Science, University of Manchester. 18
- Commision, E. (2008), 'Scl international standard classification of occupations 2008'. 142
- Dawes, J. (1991), *The VDM-SL Reference Guide*, Pitman. 12
- Dehbonei, B. and Mejia, F. (1994), *Formal methods in the railways signalling industry*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 26–34. 12, 32

- Denning, D. E. (1987), 'An intrusion-detection model', *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING* **13**(2), 222–232. 45, 48, 51
- Eades, P. (1984), 'A heuristic for graph drawing', *Congressus numerantium* **42**, 149–160. 61
- Ehrig, H. and Mahr, B. (1980), Complexity of implementations on the level of algebraic specifications, in 'Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing', STOC '80, ACM, New York, NY, USA, pp. 281–293.
URL: <http://doi.acm.org/10.1145/800141.804676> 10
- Eles, C. and Lawford, M. (2011), A tabular expression toolbox for matlab/simulink, in 'NASA Formal Methods', number 6617 in 'LNCS', pp. 494–499. 5, 15, 26, 27, 32, 82, 134, 139, 158
- English, S. and Hammond, S. (2014), 'Cost of compliance 2014', *Thomson Reuters Accelus* . 47, 49, 51
- EU (2012), Directive (eu) 2015/849 of the European Parliament and of the Council, in 'Official Journal of the European Union'. 39, 40, 41, 81, 85, 87, 88, 90, 94, 96, 106, 111, 129, 131, 132, 135, 152, 156, 162, 165
- Ezawa, K., Singh, M. and Norton, S. W. (1996), Learning goal oriented bayesian networks for telecommunications risk management, in 'In Proceedings of the 13th International Conference on Machine Learning', Morgan Kaufmann, pp. 139–147. 67, 79
- Fan, W. (2004), Systematic data selection to mine concept-drifting data streams, in 'Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', KDD '04, ACM, New York, NY, USA, pp. 128–137. 64
- FATF (2012), International standards on combating money laundering and the financing of terrorism & proliferation,, in '2016 Updates'. 36, 37, 39, 41, 83, 86, 106, 131, 152, 156

- Feng, X., Parnas, D. L., Tse, T. H. and O'Callaghan, T. (2011), 'A comparison of tabular expression-based testing strategies', *IEEE Transactions on Software Engineering* **37**(5), 616–634. 13
- Ferro, B., Suplee, C. and Flowe, J. (2015), 'Security doc roadmap', http://support.sas.com/documentation/onlinedoc/secure/9.4/Security_Doc_Roadmap.pdf [2015-01-01],. 90
- Floyd, R. W. (1967), Assigning Meanings to Programs, in J. T. Schwartz, ed., 'Proceedings of a Symposium on Applied Mathematics', Vol. 19 of *Mathematical Aspects of Computer Science*, American Mathematical Society, Providence, pp. 19–31.
URL: <http://www.eecs.berkeley.edu/~necula/Papers/FloydMeaning.pdf> 11
- Foster, D. P. and Stine, R. A. (2001), Variable Selection in Data Mining: Building a Predictive Model for Bankruptcy, Center for Financial Institutions Working Papers 01-05, Wharton School Center for Financial Institutions, University of Pennsylvania.
URL: <https://ideas.repec.org/p/wop/pennin/01-05.html> 63, 64
- Foster, D. P. and Stine, R. A. (2004), 'Variable selection in data mining: Building a predictive model for bankruptcy', pp. 303–313. 57
- Fratangelo, P. (2016), *The CDD Obligations Following a Risk-Based Approach*, Springer International Publishing. 37
- Gallier, J. H. (1986), *Logic for Computer Science*, Harper and Row. 18
- Gao, S. and Xu, D. (2009), 'Conceptual modeling and development of an intelligent agent-assisted decision support system for anti-money laundering', *Expert Systems with Applications* **36**(2), 1493–1504. 47
- Gardner, K. L. (2007), 'Fighting terrorism, the fatf way', *Global Governance* . 36
- Getoor, L. and Diehl, C. P. (2005), 'Link mining: A survey', **7**(2), 3–12. 60
- Ghosh, S. and Reilly, D. L. (1994), Credit card fraud detection with a neural-network., in 'HICSS (3)', IEEE Computer Society. 48, 64, 65

- Goldberg, H. G. and Senator, T. E. (1995), Restructuring databases for knowledge discovery by consolidation and link formation, *in* 'In First International Conference on Knowledge Discovery and Data Mining'. 50, 57, 80
- Goldberg, H. G. and Wong, R. W. H. (1998), Restructuring transactional data for link analysis, Technical report, in the FinCEN AI System."AAAI. 48, 58, 80
- Graham, S. and Lussier, D. (2012), 'Systems and methods of high availability cluster environment failover protection'. US Patent 8,117,495. 47
- Group, M. U. S. E. R. (1997), Table tool system developer 's guide, CRL Report 339, McMaster University. 2, 5, 133
- Guttag, J. (1977), 'Abstract data types and the development of data structures', *Commun. ACM* **20**(6), 396–404.
URL: <http://doi.acm.org/10.1145/359605.359618> 19
- Heitmeyer, C. (2001), Applying 'practical' formal methods to the specification and analysis of security properties, *in* 'Proc. Information Assurance in Computer Networks (MMM-ACNS 2001), LCNS 2052', Springer-Verlag.
URL: <http://chacs.nrl.navy.mil/publications/CHACS/2001/2001heimtmeier-MMM-ACNS.ps> 5, 11, 29, 32, 82, 157
- Heitmeyer, C., Archer, M., Bharadwaj, R. and Jeffords, R. (2005), Tools for constructing requirements specifications: The scr toolset at the age of ten, *in* 'International Journal of Computer Systems Science and Engineering', number 1 *in* '20', pp. 19–35. 15, 26, 27, 29, 133, 139
- Heninger, K., Kallander, J., Parnas, D. and Shore., J. (1978), Software requirements for the A-7E aircraft, NRL Report 3876, NRL. 2
- Holzmann, G. J. (1997), 'The model checker spin', **23**(5), 279–295. 29
- Horgan, J. R., London, S. and Lyu, M. R. (1994), 'Achieving software quality with testing coverage measures', *Computer* **27**(9), 60–69. 138

ISO (2002), Information technology – Z formal specification notation – Syntax, type system and semantics, Technical Report ISO/IEC 13568, International Organization for Standardization.

URL: [http://standards.iso.org/ittf/PubliclyAvailableStandards/c021573_ISO_IEC_13568_2002\(E](http://standards.iso.org/ittf/PubliclyAvailableStandards/c021573_ISO_IEC_13568_2002(E)
10

Janicki, R. (1995), Towards a Formal Semantics of Parnas' Tables, *in* 'Proceedings of the 17th International Conference on Software Engineering', IEEE, pp. 231–240.
140

Janicki, R. and Khedri, R. (2001), 'On a formal semantics of tabular expressions',
39, 189–213. 79, 106

Janicki, R., Parnas, D. and Zucker, J. (1996), 'Tabular representations in relational documents'.

URL: citeseer.ist.psu.edu/janicki96tabular.html 2, 14, 15, 19, 28, 140

Jin, Y. and Parnas, D. L. (2010), 'Defining the meaning of tabular mathematical expressions', *Sci. Comput. Program.* **75**(11), 980–1000.

URL: <http://dx.doi.org/10.1016/j.scico.2009.12.009> 20, 21, 22

Jing, M. (2000), A table checking tool, Master's thesis, McMaster. 3, 4, 15, 27, 28, 30, 32, 115, 133, 137, 139, 140

Jones, C. B. (1990), *Systematic Software Development Using VDM (2Nd Ed.)*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA. 12

Jyotindra, N. and Patel, A. R. (2011), A data mining with hybrid approach based transaction risk scoregeneration model (trsgm) for fraud detection of online financial transaction, *in* 'International Journal of Computer Applications'. 44

Kahl, W. (2003), Compositional syntax and semantics of tables, Technical report, McMaster University. 5, 133

Kim, H.-C., Pang, S., Je, H.-M., Kim, D. and Bang, S. Y. (2003), 'Constructing support vector machine ensemble', **36**(12), 2757–2767.

URL: <http://www.sciencedirect.com/science/article/pii/S0031320303001754> 63

- Kubat, M., Holte, R. and Matwin, S. (1997), Learning when negative examples abound, in 'Proceedings of the 9th European Conference on Machine Learning', ECML '97, Springer-Verlag, pp. 146–153.
URL: <http://dl.acm.org/citation.cfm?id=645325.649662> 68
- Kubica, J., Moore, A., Schneider, J. and Yang, Y. (2002), 'Stochastic link and group detection'. 58
- Lades, M., Vorbruggen, J. C., Buhmann, J., Lange, J., von der Malsburg, C., Wurtz, R. P. and Konen, W. (1993), 'Distortion invariant object recognition in the dynamic link architecture', *IEEE Transactions on Computers* **42**(3), 300–311. 45
- Lawford, M., Froebel, P. and Moum, G. (2004), Application of tabular methods to the specification and verification of a nuclear reactor shutdown system, in 'Formal Methods in System Design'. 5, 15, 27, 29, 30, 32, 45, 82, 133, 139, 145, 157
- Liu, X.-Y., Wu, J. and Zhou, Z.-H. (2009), 'Exploratory undersampling for class-imbalance learning', **39**(2), 539–550. 71
- Liu, Z., Parnas, D. L. and y Widemann, B. T. (2010), 'Documenting and verifying systems assembled from components', *Frontiers of Computer Science in China* **4**(2), 151–161. 13
- Lopez-Rojas, E., Elmir, A. and Axelsson, S. (2016), Paysim: A financial mobile money simulator for fraud detection, in '28th European Modeling and Simulation Symposium, EMSS, Larnaca', Dime University of Genoa, pp. 249–255. 75
- Maes, S., Tuyls, K., Vanschoenwinkel, B. and Manderick, B. (1993), Credit card fraud detection using bayesian and neural networks, in 'In: Maciunas RJ, editor. Interactive image-guided neurosurgery. American Association Neurological Surgeons', pp. 261–270. 67, 79
- Ngai, E., Hu, Y., Wong, Y., Chen, Y. and Sun, X. (2011), 'The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature', **50**(3), 559–569. On quantitative methods for detection of financial fraud.
URL: <http://www.sciencedirect.com/science/article/pii/S0167923610001302> 55, 56

- Otey, M. E., Ghoting, A. and Parthasarathy, S. (2006), 'Fast distributed outlier detection in mixed-attribute data sets', **12**, 2–3. 54, 55
- Owre, S. (1997), 'Using pvs in batch mode'. 140
- Owre, S., Rushby, J. M. and Shankar, N. (1992), Pvs: A prototype verification system, in 'Proceedings of the 11th International Conference on Automated Deduction', 607, pp. 748–752. 28, 29, 140
- Parnas, D. L. (1972a), 'On the criteria to be used in decomposing systems into modules', *Commun. ACM* **15**(12), 1053–1058.
URL: <http://doi.acm.org/10.1145/361598.361623> 108
- Parnas, D. L. (1972b), 'A technique for software module specification with examples', *Commun. ACM* **15**(5), 330–336.
URL: <http://doi.acm.org/10.1145/355602.361309> 11, 26, 157
- Parnas, D. L. (1992), Tabular representation of relations, in 'CRL Report'. 80
- Parnas, D. L. (1993a), 'Predicate logic for software engineering', **19**(9), 856–862. 15, 18, 19, 24, 140
- Parnas, D. L. (1993b), Some theorems we should prove, in 'HUG'93, HOL User's Group Workshop', pp. 156–163. 140
- Parnas, D. L. (2003a), 'Design through documentation: The path to software quality'. 13, 18, 44, 45, 154, 156, 159
- Parnas, D. L. (2003b), 'Documentation based software testing'. 15
- Parnas, D. L. (2006), 'Tabular expressions for software documentation', *Formal Methods McMaster Proceedings* . 16, 17, 106, 185
- Parnas, D. L. (2010), 'Really rethinking 'formal methods'', *Computer* **43**(1), 28–34.
URL: <http://dx.doi.org/10.1109/MC.2010.22> 146, 147, 148, 157
- Parnas, D. L. and Dragomiroiu, M. (2006), 'Module interface documentation - using the Trace Function Method (TFM)'. 13, 15, 23, 24, 25, 26, 33, 83, 84, 85, 106, 107, 135, 144, 148, 152, 157, 185

- Parnas, D. L., Madey, J. and Iglewski, M. (1994), 'Precise documentation of well-structured programs', **20**(12), 948–976. 15, 16
- Parnas, D. L. and Vilkomir, S. A. (2007), Precise documentation of critical software, *in* 'High Assurance Systems Engineering Symposium, 2007. HASE '07. 10th IEEE', pp. 237–244. 13
- Parnas, D., van Schouwen J. Kwan P. and S., F. (1987), Evaluation of the shutdown software for darlington (sds-1), Interim report for the atomic energy control board, Atomic Energy Control Board. 2, 14, 106
- Phua, C., Alahakoon, D. and Lee, V. C. S. (2004), 'Minority report in fraud detection: classification of skewed data', **6**(1), 50–59. 43, 44, 46
- Phua, C., Lee, V., Smith, K. and Gayler, R. (2005), 'A comprehensive survey of Data Mining-based Fraud Detection Research'. 5, 48, 49, 50, 56, 74
- Provost, F. and Fawcett, T. (2001), 'Robust classification for imprecise environments', **42**(3), 203–231. 70
- Quinn, C. (2007), Quality assessment in real-time software, Master's thesis, University of Limerick. 2, 4, 5, 10, 11, 13, 14, 24, 26, 33, 106, 133, 138, 139, 141
- Rajan, H. (2010), Building scalable software systems in the multicore era, *in* 'Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research', FoSER '10, ACM, New York, NY, USA, pp. 293–298.
URL: <http://doi.acm.org/10.1145/1882362.1882423> 47
- Riccardi, M., Soriani, C. and Giampietri, V. (2016), '8 mafia infiltration in legitimate companies in italy', *Organised Crime in European Businesses* p. 119. 36
- Rosset, S., Murad, U., Neumann, E., Idan, Y. and Pinkas, G. (1999), Discovery of fraud rules for telecommunications—challenges and solutions, *in* 'Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining', ACM, pp. 409–413. 62, 80
- Rushby, J. and Srivas, M. (1993), Using pvs to prove some theorems of david parnas, *in* 'HUG'93, HOL User's Group Workshop', pp. 164–175. 16, 157

- Russell, K. (2018), '<https://blogs.sas.com/content/sgf/2015/01/27/how-to-perform-a-fuzzy-match-using-sas-functions/>', Website. Blog Posting from SAS Technical Support. 72
- Scherrer, A. (2006), Explaining compliance with international commitments to combat financial crime: The g8 and fatf, *in* '47th Annual Convention of the International Studies Association, San Diego, CA'. 36
- Semegn, A. D. (2011), *Software architecture and design for reliability and predictability*, Cambridge Scholars. 26
- Senator, T. E., Goldberg, H. G., Wooton, J., Cottini, M. A., Khan, A. U., Klinger, C. D., Llamas, W. M., Marrone, M. P. and Wong, R. W. (1995), 'Financial crimes enforcement network ai system (fais) identifying potential money laundering from reports of large cash transactions', *AI magazine* **16**(4), 21. 47
- Serio, M. (2008), 'Politically exposed persons: Aml, taking the profit out of corruption and problems for the banks', *Journal of Money Laundering Control* **11**(3), 269–272. 39
- Shankar, N., Owre, S. and Rushby, J. (1993), 'The pvs proof checker: A reference manual'. 16, 30, 115, 157
- Siclari, D. (2016), *The New Anti-Money Laundering Law: First Perspectives on the 4th European Union Directive*, Springer. 35
- Singh, N. K., Lawford, M., Maibaum, T. S. and Wassyn, A. (2017), Use of tabular expressions for refinement automation, *in* 'International Conference on Model and Data Engineering', Springer, pp. 167–182. 134
- Singh, N. K., Wang, H., Lawford, M., Maibaum, T. S. E. and Wassyn, A. (2015), Stepwise formal modelling and reasoning of insulin infusion pump requirements, *in* V. G. Duffy, ed., 'Digital Human Modeling. Applications in Health, Safety, Ergonomics and Risk Management: Ergonomics and Health', Springer International Publishing, Cham, pp. 387–398. 31

- Sohn, S. and Seong, P. (2000), A comparative study of formal methods for safety critical software in nuclear power plant, *in* 'Journal of the Korean Nuclear Society', number 32 *in* '6', pp. 537–548. 1
- Srivastava, A., Kundu, A., Sural, S. and Majumdar, A. (2008), 'Credit card fraud detection using hidden Markov model', **5**(1), 37–48. 66
- Syeda, M., Zhang, Y.-Q. and Pan, Y. (2002), Parallel granular neural networks for fast credit card fraud detection, *in* 'Fuzzy Systems, 2002. FUZZ-IEEE'02. Proceedings of the 2002 IEEE International Conference on', Vol. 1, pp. 572–577. 65
- Tang, J. and Ai, L. (2010), 'Combating money laundering in transition countries: the inherent limitations and practical issues', *Journal of Money Laundering Control* **13**(3), 215–225. 35
- Ting, K. M. and Zheng, Z. (2003), 'A study of adaboost with naive bayesian classifiers: Weakness and improvement', *Computational Intelligence* **19**(2), 186–200.
URL: <http://dx.doi.org/10.1111/1467-8640.00219> 68
- Trancón y Widemann, B. (2014), Total functional software engineering, *in* J. McCarthy, ed., 'Trends in Functional Programming', Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 1–16. 84
- Tucker, J. V. and Zucker, J. I. (2002), 'Abstract computability and algebraic specification', *ACM Trans. Comput. Logic* **3**(2), 279–333.
URL: <http://doi.acm.org/10.1145/505372.505375> 10
- Vanasco, R. R. (1998), 'Fraud auditing', **13**(1), 4–71. 43
- Viaene, S., Derrig, R. A., Baesens, B. and Dedene, G. (2002), 'A comparison of state-of-the-art classification techniques for expert automobile insurance claim fraud detection'. 68, 79
- Wassyng, A. and Lawford, M. (2003), Lessons learned from a successful implementation of formal methods in an industrial project, *in* 'International Symposium of Formal Methods Europe', Springer, pp. 133–153. 145
- Weiss, G. M. (2004), 'Mining with rarity: A unifying framework', **6**(1), 7–19. 69

- Xu, J. and Chen, H. (2005), 'Criminal network analysis and visualization', **48**(6), 100–107. 59, 60, 61
- Xu, J. J. and Chen, H. (2003), 'Abstract fighting organized crimes: using shortest-path algorithms to identify associations in criminal networks'. 58, 60
- Yang, C. C., Liu, N. and Sageman, M. (2006), Analyzing the terrorist social networks with visualization tools, *in* 'Proceedings of the 4th IEEE International Conference on Intelligence and Security Informatics', ISI'06, Springer-Verlag, Berlin, Heidelberg, pp. 331–342. 61
- Yang, Q., Li, J. J. and Weiss, D. M. (2009), 'A survey of coverage-based testing tools', *The Computer Journal* **52**(5), 589–597. 139
- Zhang, L., Yang, J., Chu, W. and Tseng, B. (2011), A machine-learned proactive moderation system for auction fraud detection, *in* 'Proceedings of the 20th ACM International Conference on Information and Knowledge Management', CIKM '11, ACM, New York, NY, USA, pp. 2501–2504. 44