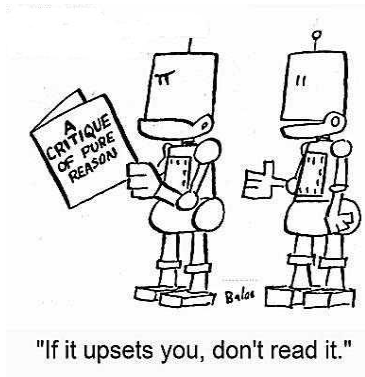


Achieving Traceable Compliance using the Ampersand Method

Master thesis Computer Science

H.W. Sangers, 838572275

September 30th 2008



"If it upsets you, don't read it."

Thoughts without content are empty,
intuitions without concepts are blind.
Kant B76

Institute: Open University of the Netherlands, School of Computer Science
Research type: Master thesis Computer Science

Title: Achieving Traceable Compliance using the Ampersand Method

Presented: September 30th 2008

Author: H.W. Sangers (Henriëtte)

Student number: 838572275

Supervising committee

Chairman: Prof. dr. ir. S.M.M. Joosten (OUNL)

Secretary: Dr. A.D. Counotte-Potman (OUNL)

Preface

As long as I can remember I was fascinated by all things and ideas in this world and how they relate. Who cares for turning cheap metals into gold if you can have understanding of it all? For some time I considered locking myself into an empty darkened room not to get out until I *understood*. Fortunately I did not for I probably still would have been there, life passing by the window.

When I got older things did not get any clearer. I saw people capturing their ideas in words where they remained like caged animals, restless, nervous, impossible to reach out and touch others. Then I entered the world of computer logic and life seemed so much easier there! You know what you start with -you can check the bits and bytes if you like, you can program what you want to end up with and you can get the mathematical proof to support it. What a relief. Would it not be soothing to live in the world of computers?

Still, the outside world was too tempting. Who can resist the wonders of this world, the beauty of art and nature, the magic of thoughts? I was left with the desire to bridge the gap between these worlds. This thesis reflects that desire within a very limited scope. After several weekends in a darkened austere room the time has now come to enter the real world again.

It is hard to finish a thesis when working full time in a challenging job. Don't let anybody tell you that it is not! To make things even harder I did just about everything to let the mission fail, apart from having children: change jobs, buy an old farmhouse, move, change jobs again. I want to thank the members of the supervising committee, Stef and Anda, for their patience and support. It took a long time to get things moving but when they did they moved fast! Also, I want to thank my family and friends for understanding when I told them I finally got serious about finishing my thesis and they would not see much of me for the months to come. Most of all I want to thank my partner Arthur, who kept pushing me not to let go. God knows at times I hated him for it but he did the right thing and I hope we can enjoy life together even more from now.

Contents

Abstract (English)	7
Samenvatting (Dutch)	9
1 Introduction	11
2 Research	13
2.1 Research questions	13
2.2 Approach	14
3 Compliance	17
3.1 Background	17
3.2 Becoming and staying compliant	17
4 The Ampersand Method	19
4.1 A problem and an answer	19
4.1.1 Mind the gap	19
4.1.2 Closing the gap	19
4.2 Introduction to the Ampersand Method	20
4.3 Why use the Ampersand Method?	22
4.4 Proof of the Pudding	22
5 Use of Ontologies in IT	27
5.1 Concept matching and ontologies in IT	27
5.1.1 A brief history	27
5.1.2 A critique	28
5.2 Overview concept- and ontology matching	28
5.2.1 Creating and using ontologies	28
5.2.2 The heterogeneity problem	30
5.2.3 Matching techniques	31
5.2.4 Matching strategies and matching systems	32
5.2.5 Processing alignments	33
5.2.6 Conclusions	33
5.3 Using ontologies to support compliance	34
6 Privacy Compliance Research at Purdue University	35
6.1 Digital Identity Management and Protection	35
6.2 Traceable and flexible compliance	36
6.3 Purdue proposal for a privacy compliant system	37
6.3.1 Matching privacy policies	37
6.3.2 Policy tracing	39
7 Using the Ampersand Method to achieve Compliance	41
7.1 Criteria for a privacy compliant system	41
7.1.1 Matching privacy policies	41
7.1.2 Policy tracing	41
7.2 A privacy compliant system using the Ampersand Method	42
7.2.1 Initial system using privacy preference templates	42

7.2.2	System with customized privacy preferences	43
7.2.3	Building flexibility into compliance	45
7.2.4	Checking loggings	47
8	Comparison Purdue and Ampersand proposals	51
9	Conclusions and Further Research	53
9.1	Ampersand, ontology matching and compliance	53
9.2	Further research	54
	Reflections on Research	55
	Bibliography	57
	Appendices	61
	Appendix 1 - ADL <i>Bankaccounts and transactions</i>	61
	Appendix 2 - Example medical ontology (OpenGALEN)	63
	Appendix 3 - Purdue code <i>Checking privacy preferences</i>	65
	Appendix 4 - Purdue code <i>Checking customized privacy preferences</i>	67
	Appendix 5 - Purdue code <i>Check logging privacy compliance</i>	69
	Appendix 6 - ADL <i>Privacy preferences using templates</i>	71
	Appendix 7 - ADL <i>Customized privacy preferences</i>	73
	Appendix 8 - ADL <i>Alternative reaction to policy</i>	77
	Appendix 9 - ADL <i>Check logging privacy compliance</i>	83
	Appendix 10 - Rules <i>Check logging privacy compliance</i>	89

Abstract (English)

Getting IT systems compliant is on top of the agenda of many organisations. Compliance means abiding by rules that apply to that type of organisation. Compliance projects currently take up a large part of IT budgets and this is not expected to change in the near future [25]. This means that organisations can benefit from methods which support compliance in IT. The *Ampersand Method* [19][20] is a promising candidate. This method uses relation algebra to capture business rules and to generate specifications and even IT systems which can be proven to satisfy these rules. If business rules can be shown to satisfy compliance demands in a certain area the Ampersand Method guarantees the IT system generated with these rules is compliant as well.

This addresses only part of the problem however. The *Compliance Survey 2007* [18] showed that translating compliance ruling into workable measures for an organisation is a difficult task for most compliance officers. Reasons for this are the nature of compliance ruling, often high level and filled with juridical terms, and the amount of ruling, coming from different regulatory entities [22]. Research shows that ontologies can be used to match concepts in compliance ruling and in organisations, both having similar domains [22][24]. This can help bridge the gap between compliance ruling and organisation specific ruling. Combining the Ampersand Method with concept matching using ontologies to capture business concepts thus becomes an interesting proposition to support compliance.

By proposing an IT environment that offers flexible and traceable compliance with privacy ruling and comparing this to a similar environment proposed by researchers at Purdue University [30], this research gives insight into the usability of the Ampersand Method to support compliance in IT systems. This research also shows whether the use of business ontologies are a worthwhile proposition when using the Ampersand Method.

The results of this research confirm the research at Purdue University by showing that creating a federated environment for automated information exchange is feasible in such a way that both traceable compliance with privacy ruling and flexibility are offered. This research also shows that the Ampersand Method offers clear advantages in adhering to business rules, which can be checked by users to establish that the desired functionality was implemented. This is important in compliance but also in other areas. Finally this research confirms the fact that using business ontologies can be beneficial when trying to achieve compliance using the Ampersand Method since these can help bridge the gap between concepts in compliance ruling and in organisations.

Further research is required to show the feasibility of matching predefined compliance rules (*compliance patterns*) to business rules in an automated or semi automated way using business ontologies. Given the fact that compliance ruling is abundant and often changes, and the fact that compliance officers find it difficult to translate compliance ruling into workable measures, supporting this would be a major improvement.

Samenvatting (Dutch)

Het compliant maken van IT-systemen staat hoog op de agenda bij veel organisaties. Compliance betekent het voldoen aan regels die gelden voor een dergelijke organisatie. Compliance projecten vergen momenteel een flink deel van IT-budgetten en dit zal naar verwachting de komende jaren niet veranderen [25]. Dit betekent dat organisaties baat hebben bij methoden die compliance in IT-systemen ondersteunen. De *Ampersand Methode* [19][20] is een veelbelovende kandidaat. Deze methode gebruikt relatiealgebra om bedrijfsregels vast te leggen en om specificaties en zelfs IT-systemen te genereren waarvan bewezen kan worden dat ze voldoen aan de gebruikte bedrijfsregels. Als van deze bedrijfsregels aangetoond kan worden dat ze compliance op een bepaald gebied garanderen dan kan de Ampersand Methode waarborgen dat het IT-systeem, dat met behulp van deze regels gegenereerd is, ook compliant is.

Hiermee wordt echter maar een deel van het complianceprobleem ondervangen. De *Compliance Survey 2007* [18] laat zien dat het vertalen van complianceregel in werkbare maatregelen voor de meeste compliance officers een lastige taak is. Reden hiervoor is de complexiteit van regelgeving, die vaak ver afstaat van de realiteit op de werkvloer en doorspekt is met juridische termen. Ook de *hoeveelheid* regelgeving van verschillende regelgevende entiteiten speelt een rol [22]. Onderzoek laat zien dat ontologieën gebruikt kunnen worden om concepten in complianceregel en organisaties, beide met vergelijkbare domeinen, te matchen [22][24]. Dit kan helpen bij het overbruggen van de kloof tussen complianceregel en bedrijfsregels. Het combineren van de Ampersand Methode met concept-matching, gebruik makend van business ontologieën, wordt zo een interessante propositie om compliance in IT-systemen te ondersteunen.

In dit onderzoek is een IT-omgeving ontworpen die flexibele en traceerbare compliance met privacyregels ondersteunt. Door deze te vergelijken met een dergelijke omgeving ontworpen door onderzoekers aan de universiteit van Purdue (V.S.) geeft dit onderzoek inzicht in de bruikbaarheid van de Ampersand Methode om compliance in IT-systemen te ondersteunen. Dit onderzoek laat ook zien of het gebruik van business ontologieën een goede combinatie vormt met de Ampersand Methode.

De resultaten van dit onderzoek bevestigen het onderzoek aan de universiteit van Purdue door te laten zien dat het creëren van een federatieve omgeving voor geautomatiseerde gegevensuitwisseling haalbaar is op zo een manier dat traceerbare compliance met privacy regels en flexibiliteit geboden worden. Daarnaast laat het onderzoek zien dat de Ampersand Methode duidelijke voordelen biedt bij het waarborgen van compliance met bedrijfsregels. Deze bedrijfsregels kunnen *direct* toegepast worden om de gewenste functionaliteit te verkrijgen en hoeven niet vertaald te worden in de gewenste functionaliteit. Verder kunnen ze gecontroleerd worden door gebruikers of auditors om vast te stellen dat de gewenste functionaliteit geboden wordt. Het onderzoek bevestigt ook dat het gebruik van business ontologieën het bereiken van compliance ondersteunt door een gemeenschappelijke set van concepten te bieden en het matchen van concepten te vergemakkelijken, en dat deze benadering goed past binnen de Ampersand Methode.

Verder onderzoek is nodig om de haalbaarheid te bepalen van het (semi)geautomatiseerd matchen van voorgedefinieerde complianceregel (*compliancepatronen*) en bedrijfsregels met behulp van business ontologieën. Gezien de grote hoeveelheid snel veranderende complianceregel, en aangezien de meeste compliance officers grote moeite hebben deze te vertalen naar werkbare maatregelen, kan dit veel organisaties grote voordelen bieden.

1 Introduction

Getting IT systems compliant and being able to prove that they are is one of the great challenges organisations are facing at the moment. What does *compliance* mean? For organisations it means abiding by laws and rules that apply to that type of organisation. Compliance concerns all employees of an organisation and all processes. Since these tend to be supported by IT systems IT systems have to be made compliant as well. Furthermore compliance needs to be *traceable* to enable organisations to show that they are compliant. Auditors should be able to 'verify' that compliance was achieved and be satisfied that the rules agreed upon were implemented correctly.

Over the past years there has been a steep increase in the number of rules organisations have to comply with. In the financial world much additional ruling was introduced like SOx, Basel-II, MIFID (Markets In Financial Instruments Directive) and CDD (Customer Due Diligence) [14]. The pace of change in this area increased the need to build compliance into IT systems in a *flexible* way, being able to adapt to new ruling on short notice and at low costs and, last but not least, being able to adapt to new business requirements as well. Being compliant after all is not the goal of organisations, it is a requirement to be allowed to do business.

So, there is our challenge: how to build compliance into IT systems in a flexible and traceable way? The Ampersand Method [19][20] seems a promising candidate to support this. This method uses relation algebra to capture business rules and generate specifications and even IT systems which can be proven to satisfy these rules. If business rules can be shown to satisfy compliance demands in a specific area the Ampersand Method guarantees the IT system generated with these rules is compliant as well. This way a *compliance certificate* could be generated, enabling organisations to show they did what needed to be done to comply with certain ruling. Furthermore, changes in compliance ruling 'only' require translating these changes into business logic and changing the business rules accordingly, after which a system that complies with the new ruling can be generated, for a moment leaving aside conversion issues.

Does this solve the problem of how to become and stay compliant? The *Compliance Survey 2007* [18] shows that translating compliance ruling into workable measures for an organisation is a challenging task for most compliance officers. Reasons for this are the nature of compliance ruling, often high level and filled with juridical terms, and the large amount of regulations from different regulating entities. At Stanford University research was done into compliance assistance systems to support the collection and analysis of relevant compliance ruling, including logic based support to check if compliance was achieved [22][24]. Ontologies are used to match concepts in compliance ruling and in organisations. Since both have similar domains this can help bridge the gap between compliance ruling and organisation specific ruling. Combining the Ampersand Method with concept matching using business ontologies thus becomes an interesting proposition to support compliance.

Now how to approach this vast field of compliance and IT systems? To get insight into the usability of the Ampersand Method to achieve compliance an established proposal for a system supporting compliance with privacy ruling is used (Purdue university [30][31]). In this research an IT environment in which federated parties can exchange data in an automated way, complying with pre-stated privacy wishes, is proposed. Ontologies are used to match privacy concepts in an automated way. Flexibility and traceability are important requirements given the personal preferences involved and the sensitive nature of privacy ruling. Proposing a similar environment using the Ampersand Method and comparing this to the

proposal by researchers at Purdue will show how the Ampersand Method can contribute to achieving traceable and flexible compliance in IT systems. Also it will show if the use of business ontologies can be a worthwhile proposition when using the Ampersand Method. This thesis shows the results of that research.

2 Research

2.1 Research questions

Compliance and IT form a very broad subject, posing many different questions. This research focuses on the usability of the Ampersand Method combined with business ontologies to achieve traceable and flexible compliance in IT systems.

The reason to do research into compliance and IT is that compliance demands currently are on top of the agenda of many organisations and compliance projects absorb a large part of IT budgets. This situation is expected to last for quite some time [25]. This means that there is much to be gained for organisations by finding ways to support compliance in IT [23]. The Ampersand Method combined with the use of business ontologies is a promising candidate. Using *Ampersand* all business logic can be found in one place, making it easier to verify whether compliance requirements are met and facilitating changes when compliance ruling changes. Also, compliance rules are used *directly* to generate IT systems instead of having to program business processes leading to compliance. Mathematical proof can be given that the system generated complies with the business rules used to generate it [19][20] (this research builds upon the results achieved by these researchers and giving this proof is not in scope of this research). *Business ontologies* can be used to bridge the gap between compliance ruling and business concepts. The use of business ontologies fits in well with the Ampersand Method which uses ontologies itself. This makes it worthwhile to check what the combination of these approaches can offer to achieve compliance in IT.

Recent developments in the area of compliance and IT systems are described in articles by amongst others researchers at Stanford [22][24] and Purdue university [3][4][5][30][31]. The articles by researchers at Stanford university concentrate on the development of compliance assistance and decision support systems. The articles by Purdue researchers concentrate on achieving traceable and flexible compliance with privacy ruling. The Purdue research offers an interesting *case study* [30]. An IT environment in which federated parties can exchange data in an automated way, complying with pre-stated privacy wishes, is proposed. By proposing a similar IT environment using the Ampersand Method and ontologies this research will try to reach conclusions about the usability of Ampersand in this area.

The research questions of this thesis, following from the issues described above are:

1. Is it possible to create an IT environment (or functional specifications for this) using the Ampersand Method, which provides traceable and flexible compliance with privacy policies, as described in the article by researchers of Purdue University [30]?
2. How does this environment compare to the one proposed by the researchers at Purdue and what does this tell us about the usability of the Ampersand Method to achieve compliance?

Next to looking into what the Ampersand Method can add to achieving compliance in IT systems this research has added benefit by checking if the way to match concepts in compliance rules with business concepts, as proposed in recent articles, fits in well with the Ampersand Method. Business ontologies are key in this. This leads to the next research question:

3. Can the use of business ontologies facilitate achieving compliance when using the Ampersand Method?

2.2 Approach

This research is not part of an IT business assignment. Combining this research with an IT business assignment could divert attention from the more academic issues at stake. The research was performed in the academic surroundings of the Dutch Open University. Researchers involved in the reference research at Purdue university were contacted for input and feedback.

In this thesis extra attention is paid to two subjects, which are fundamental for this research:

1. *Ampersand Method*

The Ampersand Method is the method of choice in this research. In chapter 4 an elaboration on this method based on several articles and books can be found, and an example of how to use this method.

2. *Ontology matching as used in IT systems*

Concept matching using business ontologies is combined with the Ampersand Method and thus is fundamental in this research as well. In 2007 a book was published called *Ontology Matching* [12] on the use of concept matching and ontology matching in IT. This book was used for this research, next to several articles. More on this subject can be found in chapter 5.

Research model

The research model that was used in this research can be found in figure 1. The research consisted of the following steps:

1. Analysing the article on traceable compliance from Purdue University [30].
2. Getting up to date on compliance and IT, the Ampersand Method and the use of ontologies in IT.
3. Contacting researchers who wrote the article to get more information and feed back.
4. Creating an environment similar to that in the Purdue article using the Ampersand Method.
5. Analysing the differences between the Purdue environment and the Ampersand environment.
6. Drawing conclusions on the usability of the Ampersand Method combined with business ontologies compared to the proposal by Purdue researchers. This could lead to different insights:
 - a. Using the Ampersand Method gives comparable results: this verifies conclusions of the Purdue researchers.
 - b. Using the Ampersand Method does not give the desired results: this may lead to conclusions about the research at Purdue or about the usability of the Ampersand Method in this context.
 - c. Using the Ampersand Method gives better results: this verifies conclusions of the Purdue researchers and will lead to positive conclusions about the usability of the Ampersand Method in this context.

7. Finalising thesis.

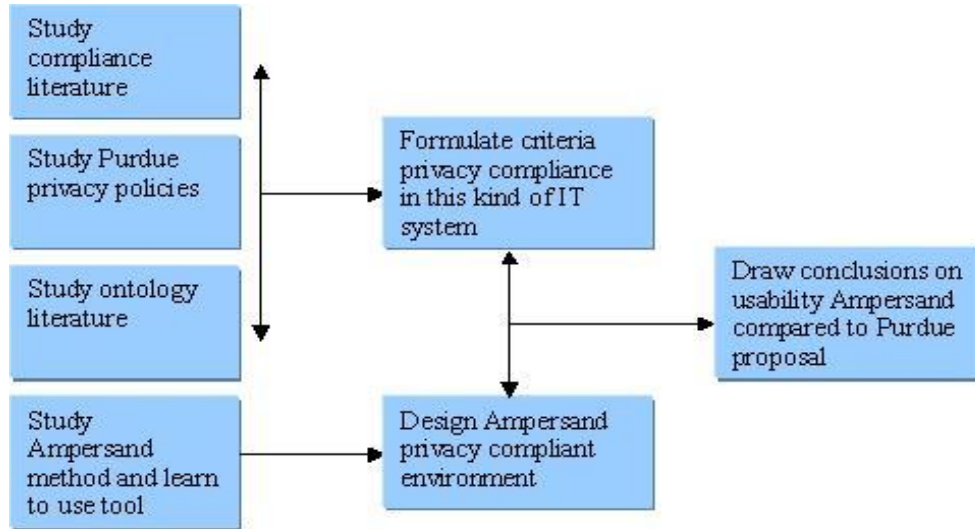


Figure 1: Research Model

Composition of thesis

Current developments in compliance and IT are described in chapter 3. In chapter 4 the Ampersand Method is elaborated upon. The reasons to use this method are described. After this an introduction to the theory behind the method is given, and an example of the use of the method is provided.

In chapter 5 concept matching and the use of ontologies in IT are described. First some attention is paid to the history of ontologies in IT, and to the roots of the phenomenon of *ontology*, in philosophy. The current use of ontologies in IT is described, and new developments. Special attention is paid to the problem of *matching ontologies*, fundamental when confronted with the use of different ontologies by different parties. Finally the use of concept matching and ontologies to achieve compliance is described.

In chapter 6 the research into privacy compliance at Purdue University is introduced, and the Purdue proposal is presented. In chapter 7 the criteria for a privacy compliant IT system as described in chapter 6 are analysed and an alternative system using the Ampersand method is proposed. In chapter 8 both proposals are compared.

In chapter 9 the conclusions on the usability of the Ampersand Method and business ontologies to achieve traceable and flexible compliance in IT systems are presented. Further research is proposed to elaborate on the results achieved.

3 Compliance

3.1 Background

Compliance, as used for organisations having to operate according to certain rules, goes back to the nineteenth century when in the USA and in the UK laws were made to regulate businesses. An early example is the UK Companies act of 1862 in which the management of companies limited by shares was arranged [23]. Concentrating on the financial world the severe economic crisis of the nineteen thirties made clear that strong interventions were needed, leading to more regulations and the foundation of the BIS, the Bank for International Settlements in Basel, a regulatory based institution meant to increase international financial stability [14].

The financial world expanded quickly in the twentieth century and regulations fell short. In recent years scandals like Enron, Parmalat and the downfall of the Barings Bank lead to a substantial increase in compliance ruling. Public trust in financial institutions is crucial for the economy and this trust had to be restored by new regulations [14]. Being compliant is now expected of all organisations, though the amount of compliance ruling differs per type of organisation.

3.2 Becoming and staying compliant

Most people will agree that regulating businesses to some extent is necessary, but in practice for many organisations compliance is quite a burden. There are several reasons for this, amongst others:

- Compliance ruling is often complicated, referring to different standards which usually are not easy to understand either [24].
- There is a lot of compliance ruling, from different legal and professional entities. The sheer amount makes it hard to get the complete picture. In the worst case the ruling is not consistent, even contradicting [22].
- Translating compliance ruling into workable measures for an organisation is found to be very difficult by most compliance officers [18].
- Compliance measures have to be implemented in processes, procedures and IT systems, all bringing along their own challenges [17].
- Auditors need to be satisfied that compliance is achieved so the implementation has to be traceable [17].

Some important issues pop up: the difficulty to translate compliance ruling into workable measures, the need to adapt IT systems and traceability, being able to 'prove' compliance.

The problem to *translate compliance ruling* is caused by the fact that there are so many regulations, sometimes even contradicting, that businesses are having a hard time translating all relevant compliance ruling into workable measures [22][24]. Also, compliance officers usually do not have a legal background, making it difficult to understand all legal requirements. Researchers at Stanford University developed a compliance assistance system to try and tackle these problems [22]. This system collects all relevant ruling in a certain area, matching concepts using ontologies and using logic to test whether compliance was achieved. These systems are experimental however, and they are not available to most businesses.

Even regulatory institutes realise it is not possible to capture every possible situation in regulations. In recent years the focus shifted from a *rule based* to a *principle based* approach [2]. In a principle based approach the *intention* of the ruling is crucial, in a rule based approach the *rules themselves* are. The latter approach requires more ruling and may lead to the conclusion that everything is allowed as long as there is no rule against it. The first approach gives people more freedom to decide how to act as long as it is in the spirit of the ruling, holding them accountable for their actions when they go against it. In this situation we find more *a posteriori* checking, making adequate logging necessary [8].

Adapting IT systems is essential to achieving compliance since IT systems in most businesses form the heart of the administrations. A survey by Mercury showed that businesses expect a large amount of their IT budget will go to compliance projects in the next years [25]. Research by Deloitte and Touche showed that *complexity of IT environments* is seen as a major impediment in compliance projects [1]. The existence of large numbers of legacy systems and lack of control over IT maintenance prevent organisations from taking efficient action. IT Governance plays an important role in achieving compliance [25]. A strong IT framework is needed, and compliancy should be part of this. On the other hand the need for compliance can also increase IT efficiency and clean up procedures. Research by Gartner showed that companies which react separately to every new regulation spend much more on compliance than organisations which react proactively to new ruling [23]. Some organisations experienced a competitive advantage handling these issues in a more efficient way than others.

Traceability is an inseparable part of compliance. It is not enough for a business to state that it is compliant, this has to be verified by auditors. Auditors initially came from the ranks of business administrators. When IT became more important for businesses EDP auditing¹ or IT auditing became a separate profession. It is expected that these professions will gradually merge since nowadays it is not possible to audit a business administration without basic knowledge of IT, nor is it possible to audit IT systems without basic knowledge of business administration [17]. These auditors have to establish that compliance was achieved.

In IT the *compliance challenge* in the coming years will be to adapt IT systems in such a way that their compliance is traceable, increasing rather than deminishing systems flexibility since business demands and compliance ruling will certainly change [23]. Also, to leave room for people using the systems to make their own decisions in line with the intention of regulations, logging their actions so they can be made accountable [8]. Systems development methods that offer advantages in these areas will have added value for many businesses.

¹Electronic Data Processing

4 The Ampersand Method

4.1 A problem and an answer

4.1.1 Mind the gap

Though few people will deny IT systems largely increased the quality of our lives it has been clear from the beginning of IT times that there is a gap between what people want and what IT systems deliver. To get an IT system to do exactly what you want takes two things: first you have to make clear what you want, second you have to implement this in a system.

Now what is the problem? Making clear what you want does not sound too complicated, but it is! People to a large extent live their lives *on the fly*. They do not think upfront about everything that might happen and how they want IT systems to react to that. They know what they want when a situation arises, but IT systems need to be programmed long before this. So we need people who investigate what is needed in situations that might arise long before they do, let us call them analysts. These analysts talk to future users to get a clear picture of the desired functionality. One issue that will certainly come up during this exercise is the fact that computers work along strict logical lines. If A happens do B. If we want a computer to react in different ways to situation A we have to specify this: if A happens then check if C is true, if so do B, else do D. People usually do not express their wishes like this. They may say in case of A the system should do B, and later, thinking of a different context, say in case of A the system should do D. These kind of inconsistencies have to be resolved and it has to be made clear when a system should react to A with B, and when with D. All this needs to be written down in such a way that people who implement these wishes know exactly what is meant. Lack of clarity in this stage causes the first gap in IT systems development.

We now have specifications and we can start implementing them. Even if the specifications are written down in an unambiguous way they could be wrongly implemented. Programming, after all, is mostly done by humans. Humans, *even* IT staff, do not think the way computers operate. Apart from this, manual work is by its nature error-prone. This accounts for the second gap between what people want and what IT systems deliver.

4.1.2 Closing the gap

The gap between what people want and what IT systems deliver contributes to the bad track record of IT projects. Many attempts were made to close this gap. Formal modelling languages were developed, like Z^2 , using set theory and first order predicate logic to specify IT systems. These formal modelling techniques have a strong mathematical basis and fit in well with the way computers operate. It is difficult to match them with informal human thinking however [10].

Closer to the way humans think are informal modelling techniques like **the UML** [29], developed by the Object Management Group (OMG). The UML consists of a number of proven modelling techniques like use cases and sequence diagrams. These can be used to discuss specifications with future users. Many models are *visual*, enabling users to identify with them but also lacking formal precision needed to specify IT systems. The precise UML group, pUML³, tries to fill this gap by providing a formal basis for the UML. An important development was the addition of the *Object Constraint Language* (OCL) to the

²<http://vl.zuser.org/>

³<http://www.cs.york.ac.uk/puml/>

UML, extending possibilities to add constraints to models. The UML originally was not developed to formalise specifications however, rather to specify, visualise and document. Even though *to provide a formal basis for understanding the modeling language* was added as a goal in the UML specifications the OMG itself declares that full formal specification of the UML is not realistic⁴. Attempts to generate IT systems using UML specifications show inconsistencies in the UML metamodel, making it impossible to specify unambiguously [13]. Also, the UML metamodel was found to be too complex to be used in practice for this goal [26]. The fact that full formal specification of the UML metamodel is not available makes it difficult to discuss the correctness of models in a formal way [27]. This limits the usability of the UML to generate IT systems.

The **Ampersand Method** can be viewed as an attempt to close the gap between informal human specifying and the formal way computers operate. This method facilitates getting clarity about specifications and makes it possible to derive functional specifications for IT systems directly from business requirements [20]. These functional specifications can be used to build or even generate an IT system that can be proven to implement the business requirements correctly [19][20].

How does this work? Business requirements are captured in business rules. Business rules tell which values should be upheld at any time. These business rules come from the business and can be understood by the business, supporting the process of getting clarity about desired functionality. Furthermore, using the Ampersand Method there is no need to program the business processes to uphold these rules, the rules are the invariants of the system that is generated. To be able to generate specifications based on business rules these rules have to be formally sound, leaving no room for different interpretations. To ensure this the language ADL⁵ was developed. This language is based on relation algebra but is set up in such a way that no in-depth knowledge of this subject is required. Analysts can translate business requirements into business rules specified in ADL. These business rules are then used to create functional specifications in an automated way. So, business rules are not added to a predefined IT system, the IT system is derived from the business rules. This way, if the business rules are correct, the IT system can be proven to be correct as well [19]. This makes the Ampersand Method an interesting candidate to close the gap.

4.2 Introduction to the Ampersand Method

The Ampersand Method is based on the Calculating with Concepts technique [11][21], which was originally developed to improve the precision of UML class diagrams and allow formal reasoning about them [10]. It was discovered however that this technique could also be used to implement business rules and generate IT systems that fully align with business logic. A language was developed in which to express business rules, ADL, and a tool was developed to generate specifications and even IT systems based on these business rules. This and more resulted in the Ampersand Method [19][20].

The Ampersand Method uses *relation algebra* to implement business logic in IT systems. Relation algebra is a well established mathematical area which has been studied for a long time, amongst others by De Morgan (1883). Its application in computer science dates from several years back. Brink's book *Relational Methods in Computer Science*[6] shows how relation algebra, set theory and logic are related and can be used to produce formally sound IT systems. Formal logic is used to ensure the correctness of the implementation of business

⁴<http://www.omg.org/>

⁵'A Description Language'

logic. Relation algebra is equivalent to predicate logic, but is easier to use [19].

The building blocks used in Ampersand are: *concepts*, *relations* and *rules*.

1. Concepts

Concepts are entities that are of importance to users, like patients in a hospital, or customers and products in a shop. Concepts are defined within a *context*, which could be a shop. Between concepts there can be relations.

2. Relations

In *relation algebra* relations and operators are key. Operators are used on one or more relations to derive other relations. In Ampersand relation algebra is used with relations operating on *sets* (concepts). A relation between the concepts product and customer can be that a product is bought by a customer.

Properties of relations, like *multiplicities*, can be used to capture business requirements. Relations can be [19]:

- *univalent* [in ADL: UNI]
A relation $r: AxB$ is univalent if each instance of A corresponds to at most one instance of B.
- *total* [in ADL: TOT]
A relation $r: AxB$ is total if each instance of A corresponds to at least one instance of B.
- *injective* [in ADL: INJ]
A relation $r: AxB$ is injective if each instance of B corresponds to at most one instance of A.
- *surjective* [in ADL: SUR]
A relation $r: AxB$ is surjective if each instance of B corresponds to at least one instance of A.
- *function* [in ADL: $- >$]
A relation $r: AxB$ is a function if it is both univalent and total.
- *bijective*
A relation $r: AxB$ is bijective if it is univalent, total, injective and surjective.

Qualifying the relation between product and customer as a function means that each product is bought by a customer and that a product can not be bought by more than one customer. If this rule is broken the system generated with these specifications will report this and the problem can be solved.

3. Rules

Rules are the invariants of the system. These will be upheld no matter what, and if they capture the business logic this means that business logic will be upheld by the system. A rule could be that a bill can only be sent after a product was bought by a customer. That implies that, before sending a bill, the system needs to verify that the product on the bill was bought by a customer.

So, in Ampersand there are concepts, relations between these concepts and business rules in which business logic is captured. There is a language to express business rules in, ADL, and a tool to generate system specifications or even IT systems based on these concepts, relations

and rules. To achieve this relation algebra is used, which can provide mathematical proof that the system aligns with the business logic as stated in the business rules and relations [19][20].

4.3 Why use the Ampersand Method?

There can be many reasons to use the Ampersand Method. The Ampersand Method, using formal specification of business logic, leaves no room for inconsistency. That means Ampersand can be used to resolve inconsistencies in functional specifications [19]. Users may have different views on which business rules apply. Also users may not be aware of consequences of their own rules and inconsistencies among these rules. For instance: in a bank a rule might be that only someone with an account with the bank is considered a client. Another rule might be that an account can only be set up for an existing client. Combining these rules means the system gets stuck. It is impossible to set up a new client since at the moment of creation he does not have an account yet and it is impossible to set up an account for someone who does not yet exist as client. Confronting users with this inconsistency will lead to improved specification of business rules. Users may decide it should be possible to set up an account for a prospect client, or to set up a client provided he is linked to an account in the same session. This is a simple example that most analysts can tackle without using the Ampersand Method, but in a complex system with many business rules inconsistencies can easily be overlooked. The Ampersand Method can be used to derive unambiguous system specifications that suffice to develop IT systems even off-shore, if desired. A technique which can be considered part of the Ampersand Method and which can be used to improve specifications is *cycle chasing* [11]. Different relations between concepts are used to enforce business logic. We can reverse this approach by looking for different paths between concepts, forming *cycles*, and see if we can derive business logic from that. This can help to create awareness among users about implicit business logic or about consequences of business rules.

The full strength of the Ampersand Method is in its potential to detect violations of business rules and act accordingly however [20]. Using Ampersand we can generate an IT system that can be proven to align fully with business logic as captured in relations and rules [19][20]. This closes the second gap in IT development, caused by incorrect implementation of correct specifications, the first gap being narrowed by Ampersand supporting to get clarity about desired functionality. Being able to look at business logic and know that what you see is what you get can save a lot of testing and lengthy discussions.

All areas involved in IT will see the benefit of this, but it provides especially interesting features for *compliance*. If we can determine that a set of business rules satisfies compliance requirements in a specific area, an IT system developed with the Ampersand Method using this set of business rules can be proven to be compliant. There is no need to program the business processes that should govern compliance, leaving room for misconceptions when implementing the logic, the compliance rules are upheld as invariants of the system [21]. Also, using Ampersand all business logic can be found in one place, making it easier to verify whether compliance requirements are met, and facilitating changes when compliance ruling changes. Both traceability and flexibility are thus enhanced.

4.4 Proof of the Pudding

The proof of the pudding is in the eating, so let us see how the Ampersand Method turns out in practice. A simple **example** from a banking environment: a bank has clients, accounts

and transactions. An account belongs to a client, and a client can do transactions taking money from an account. The ADL program to express this can be found in annex 1. The ADL code can be processed using an online compiler⁶. Access can be given by Prof. Joosten (see: *supervising committee*).

The ADL program *Bankaccounts and Transactions*:

- Within the *context* Bank the *concepts* client, account and transaction are defined.
- Within the *pattern* Transactions the *relations* between client, account and transaction are defined. All relations are functions, meaning every concept A is related to exactly one concept B. So, every account belongs to exactly one client, every transaction is ordered by exactly one client and so on.
- After defining relations *rules* can be specified to make sure business logic is upheld. In this case it is specified that transactions can only be ordered by the owner of the account the money is taken from.

With this short ADL program specifications can be created including a datamodel and services. Also an online system can be created, with tables populated with the data found in the specifications. Transactions are registered, including the client who ordered them and the account the money is taken from. Clients, accounts and transactions can be added. More interesting, violations of business logic are signalled and need to be resolved. Now let us taste the results!

Functional Specifications

Functional specifications were made for this banking example by running the ADL program in annex 1 using the online ADL compiler mentioned above. This results in elaborate specifications presented in a 13-page PDF file. Some parts of these specifications are discussed below.

After an introduction the specifications start with the agreements that were made about Transactions, the pattern that was specified within the context Bank. In figure 2 you can see the agreements as specified. NB: part of the text is in *Dutch*, the language produced by the compiler that was used. The agreements represent both the business rules (rule 1: A client can only order a transaction that takes money from his own account) and the relationship properties (rule 2: relationship concerned is bijective, rules 3-4: relationships concerned are functions). These agreements are in 'normal' language and can easily be discussed with users to make sure that they are correct and complete. If not, rules can be changed or added.

1. A client can only order a transaction that takes money from his own account.
2. Elke account belongs to precies één client en vice versa.
3. Elke transaction is ordered by precies één client.
4. Elke transaction takes money from precies één account.

Figure 2: Agreements about transactions

⁶<http://86.88.190.85/ADL.php>

In figure 3 you find an elaboration of the one business rule used in this example. All relationships involved are mentioned, and the invariant: if a transaction is done by client A the account from which money is taken must belong to client A.

1	<p>A client can only order a transaction that takes money from his own account.</p> <p>Relaties:</p> <p style="padding-left: 40px;"><i>isDoneBy</i> : <i>Transaction</i> × <i>Client</i></p> <p style="padding-left: 40px;"><i>takesMoneyFrom</i> : <i>Transaction</i> × <i>Account</i></p> <p style="padding-left: 40px;"><i>belongsTo</i> : <i>Account</i> × <i>Client</i></p> <p>Regel:</p> <p style="padding-left: 40px;">$\forall c :: Client; a :: Account; t :: Transaction :$</p> <p style="padding-left: 40px;">$c \ I \ isDoneBy(t) \wedge \ takesMoneyFrom(t) \ I \ a \Rightarrow \ a \ belongsTo \ c$</p>
---	---

Figure 3: Transaction rules

In figure 4 you see an example of the data structure analysis performed. The three concepts used in this example are specified, including their relations and services to create, update, delete and call them. All parts of the system are specified in this way, with further elaboration of relations and properties.

Client	Transaction	Account
<p>+ belongsToInv : Account</p> <p>+ newClient(belongsToInv):handle</p> <p>+ getClient(handle):[belongsToInv]</p> <p>+ delClient(handle)</p> <p>+ updClient(handle,belongsToInv)</p>	<p>+ isDoneBy : Client</p> <p>+ takesMoneyFrom : Account</p> <p>+ newTransaction(isDoneBy,takesMoneyFrom):handle</p> <p>+ getTransaction(handle):[isDoneBy,takesMoneyFrom]</p> <p>+ delTransaction(handle)</p> <p>+ updTransaction(handle,isDoneBy,takesMoneyFrom)</p>	<p>+ belongsToFun : Client</p> <p>+ newAccount(belongsToFun):handle</p> <p>+ getAccount(handle):[belongsToFun]</p> <p>+ delAccount(handle)</p> <p>+ updAccount(handle,belongsToFun)</p>

Figure 4: Data structure

In figure 5 you see a specification of a service as can be found for all services identified (mentioned in figure 4). The service newTransaction creates a Transaction. The behaviour of the service when called is described as well. These specifications can be handed over to IT staff to be implemented as part of the new system.

```

newTransaction

newTransaction( In   i : ClientHandle   ;
                In   t : AccountHandle  ;
                Out  obj : TransactionHandle )

Bij aanroep gedraagt deze service zich als volgt:

newTransaction(i,t,obj)
{Post: obj.isDoneBy = i      and
      obj.takesMoneyFrom = t }

```

Figure 5: Services

Finally, in figure 6 you find a Function Point Analysis, describing the complexity of the services belonging to the concept Transactions. A calculation is made of function points for

the different services and for Transactions as a whole. These calculations are made for every part of the system and can be added up to get an indication of the effort needed to build the system as a whole. Depending on how the system is built function points can be translated into *hours needed* or *budget needed*. NB: this feature is used to illustrate the possibilities of the Ampersand engine, the correctness of this analysis was not tested in this research.

FPA complexiteit:

concept	type	fp
newTransaction	IF Gemiddeld	4
getTransaction	OF Eenvoudig	3
delTransaction	IF Gemiddeld	4
updTransaction	IF Gemiddeld	4

Transaction als geheel is 22 functiepunten waard.

Figure 6: Function Point Analysis

The specifications generated with the ADL program can be used to discuss business logic with users and, after they agree, they can be handed over to IT staff to build the system. Another option is to generate a system directly with the ADL program using the ADL compiler. This option is described in more detail below.

Generating a system using ADL

In figure 7 you find an example of the interface of the system that was generated using the ADL program in annex 1.

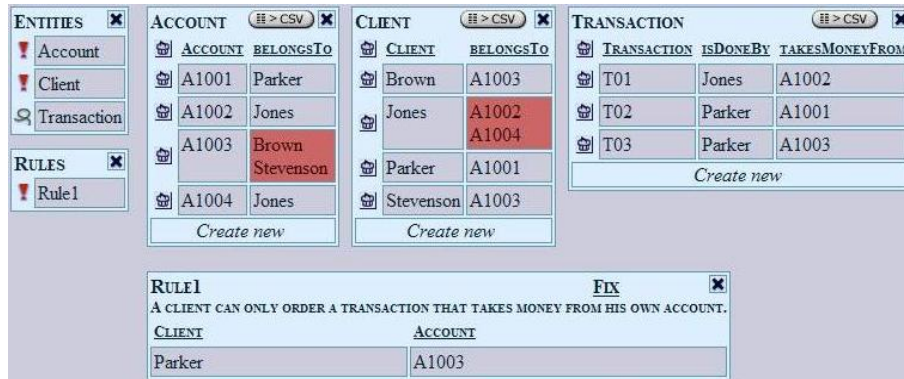


Figure 7: Screen Bank

In this example transaction T03 is ordered by Parker, but transfers money from an account of Stevenson and Jones. This violates *rule 1* of the program. After processing we get the message that rule 1 is broken because Parker is not allowed to do a transaction on account A1003, which is the account belonging to Stevenson and Jones. You can see this in the lower half of the screen.

Different steps can be taken to solve this problem. Parker may have entered the wrong account number. If the account number connected to the transaction is changed to A1001 the warning disappears. Parker now does a transaction on his own account. The warning also disappears if the person who did the transaction is changed to Stevenson or Brown, the owners of account A1003. So, different steps can be taken to reconcile the data in the system, depending on business demands.

Other ways to enforce business logic are used as well, like properties of relations. By specifying the relation between account and client is a *function* no account may belong to two clients. The fact that account A1003 belongs to both Stevenson and Jones violates this rule. In the system created we are notified of this violation and we need to solve it. Another restriction could be that a client can have at most one account. To achieve this the property *injective* is added to the relation between account and client. This creates the warning that Jones has two accounts. The correctness of the implemented functionality was established by extensive testing of the system that was generated.

It is for the business to decide which logic they want to apply, Ampersand can be used to create a system that aligns with this logic.

5 Use of Ontologies in IT

5.1 Concept matching and ontologies in IT

5.1.1 A brief history

Ontologies have drawn considerable attention in information analysis and systems design over the past years. Information systems tend to be seen as a *representation of the real world* [32]. Ontologies provide a vocabulary to describe this world and they show how different concepts are related [12]. The building blocks of ontologies are sets (or concepts), attributes (or properties), and relationships. The definitions of these building blocks include information about their meaning and about constraints on their logically consistent application. Ontologies are typically specified in languages that allow abstraction away from data structures and implementation strategies, for this reason ontologies are said to be at the *semantic* rather than at the *physical* level⁷.

The history of IT systems development is filled with misunderstandings about concepts [32]. If it is not clear what is meant with, for instance, a *transaction*, how can a system be built to administrate them? Even within organisations different departments use concepts in different ways. In a bank, *loss* in the financial department can be quite different from *loss* in the risk department. To be able to capture both in IT systems the difference between them and their relation need to be made clear. Ontologies support this. In recent years the internet added a dynamic dimension to the use of ontologies in IT.

The interest in ontologies in IT started within Information Systems Analysis and Design (ISAD). This discipline felt the need to capture concepts in the real world and match these with concepts in information systems design [32]. Also, different modeling techniques were developed over the years and these had to be compared using well founded criteria. A familiar attempt at this is the Bunge-Wand-Weber (BWW) representation model, which provides a theoretical base to evaluate information systems modeling techniques. The premise used for this model is that a modeling technique should be able to represent all things in the real world that could be important to users, otherwise it is not complete [15]. Wand and Weber studied the philosophical branch of ontology, meta-physics, and extended the systems ontology developed by Bunge⁸. They analysed the ontological completeness and ontological clarity of modeling techniques to discover their strength and weaknesses.

Green and Rosemann, in their article on integrated process modeling [15], use the BWW model to analyse the five views essential in information systems development: proces, data, function, organization and output. They use this to evaluate CASE tools and ERP systems.

A new dimension was added to the ontology matching problem by developments like agent technology and ambient intelligence and by the rise of the internet. In Artificial Intelligence agents or software entities need shared concepts to cooperate. One way to achieve this is by letting these entities commit to a shared ontology [16]. A more complicated but also more flexible alternative is to merge the different ontologies dynamically.

Better known to most people are internet shops, which today are as well known as high street warehouses. We use portals that connect with numerous suppliers of goods. Matching demand and supply in this dynamic environment is a challenging task. How do we match *couch* with *sofa*, *mansion* with *villa*? By using ontologies! Different ontologies are matched dynamically, enabling *common understanding* of concepts [12]. The internet has increased the interest in ontologies in many organisations.

⁷<http://tomgruber.org/writing/ontology-definition-2007.htm>

⁸<http://www.dooy.salford.ac.uk/ext/bunge.html>

5.1.2 A critique

A well established but complex branch of philosophy and the virtual world of the internet: an interesting match. Some people consider ontologies to be *the silver bullet*⁹ for knowledge intergration, but are they? Using ontologies in itself does not reduce heterogenity, it just raises the problems to a higher level [12]. When everybody has his own ontology how does this help us to develop a common understanding?

According to Wyssusek [32] the mere term *ontologies* reveals our misunderstanding of ontology. After all, we do not speak of *biologies* to acknowledge different views of biologists. There is only one science called *biology*, with people having different views on it. *Ontology* as used in IT refers to the *construct*, the linguistic convention used by a party to describe the world, not to the philosophical discipline of Ontology. Most IT people are not philosophers, though using the knowledge gained in other disciplines can be very fruitful there is a clear danger of misinterpretation, especially in a branch like philosophy where terms need to be understood within the context of historic debates [32].

Ontology, as used in philosophy, is linked to the very essence of existence. To answer the question *what exists* we have to determine what *it means* to exist. Kant (1787, B303) rejected ontology for claiming to supply synthetic a priori knowledge of things in general. Not even proof for the existence of things outside of us had yet been given so how could this knowledge be justified? Heidegger (1962) took this one step further, claiming not so much this lack of proof but the fact that this proof is still expected is the real problem. Wittgenstein (1922) concluded that *the limits of our language mean the limits of our world* [32]. We construct our own reality using language, which brings us back to everybody having his own ontology, not only to express but to *create* his own world. Ontology matching is used to bring these worlds together. What we lost on the way is the claim to universal truth, ontology representing the essence of existence. That does not make ontology matching less useful however.

5.2 Overview concept- and ontology matching

5.2.1 Creating and using ontologies

Why would someone want to use and even develop an ontology? Some of the reasons are [28]:

- to share understanding of the structure of information among people or software agents
- to enable reuse of domain knowledge
- to make domain assumptions explicit
- to separate domain knowledge from operational knowledge
- to analyse domain knowledge

Creating ontologies is a daunting task. Ontologies have to be logic and consistent, and they usually cover a substantial domain. There is no *one correct way* to model a domain [28], making it difficult to know whether you are on the right track to get the results you want. An ontology is a construct, reflecting the way of thinking of its creator. Since there are quite some ontologies available already it is worth looking for an ontology that matches your needs before starting to create one yourself. Reusing an ontology can also help conforming to

⁹*Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce* by Dieter Fensel

existing standards. There are libraries of ontologies on the internet, for instance Ontolingua¹⁰ or DAML¹¹.

If one does decide to develop an ontology the steps to take would be [28]: defining classes, arranging these in a subclass-superclass hierarchy, defining properties and allowed values for these and filling in the values of properties. Different approaches are possible. One could go for a top-down or a bottom-up approach. In a top-down approach the first step would be to define the highest level classes in the domain, for an ontology about living beings this could be: animals, plants and so on. After this subclasses would be identified: for animals mammals, fish. This approach would be followed all the way down, attaching the instances to the lowest level. In a bottom-up approach one could start with a list of all instances one can think of, grouping these into subclasses. These subclasses would again be grouped into higher level classes all the way up. In practice often a mix of these approaches is taken, starting with a number of instances and some superclasses, gradually defining the middle layer and refining the upper- and lower levels. More about developing an ontology can be read in *Ontology Development 101: a Guide to creating your first Ontology* [28].

After looking at how to create an ontology let us look at their applications ([12] Ch1).

- *Information integration*

Information integration covers different problems like schema integration, data warehousing and catalogue integration. *Schema integration* is one of the oldest scenarios for the use of ontologies. If businesses merge different IT systems covering the same domain often have to be integrated, leading to schema integration. The best known examples of *catalogue integration* are sales portals like eBay and Amazon. Other examples are large-scale product classifications like UNSPSC, the United Nations Standard Products and Services Code¹². Information from different sources may even be integrated in a virtual layer, giving immediate access to underlying data sources. This is called *data integration*.

- *P2P information sharing*

Peer-to-peer (P2P) refers to a distributed communication model in which peers have equivalent roles in providing and using data and services. A well known example is Kazaa, used to exchange music and other files. To establish meaningful information exchange user ontologies have to be matched. To avoid this sometimes schemas are imposed on users. The constant matching of ontologies may lead to *emergent semantics* between peers, meaning that peer ontologies gradually converge towards a consensus ontology.

- *Web service composition*

Web services are processes that can be evoked by users through the internet. To enable users to find the services they want and integrate them the different ontologies used need to be matched. This is done by a *data mediator*, either offline designing a service composition, or online requiring a dynamic approach.

- *Autonomous communication systems*

To enable multi-agent communication or to match contexts in ambient computing,

¹⁰<http://www.ksl.stanford.edu/software/ontolingua/>

¹¹<http://www.daml.org/ontologies/>

¹²<http://www.unspsc.org/>

where services are used provided by the surrounding environment, a common understanding is needed in which ontologies can play an important role [16].

- *Navigating and query answering on the web*
Navigating on the web, using queries to get the information we want, is daily routine for most of us. To select and aggregate the information derived from multiple heterogeneous sources using their own ontologies ontology matching is key. Much of this is done on the fly, requiring a very dynamic approach.

5.2.2 The heterogeneity problem

Heterogeneity among concepts and ontologies creates many problems but is unavoidable in distributed and open systems like the internet. First let us take a look at the different forms an ontology can take. These range from simple glossaries, which are mere lists of terms, through taxonomies and metadata models to logics and formal ontologies, in increasing order of formality and expressiveness. Ontologies are presented in an ontology language like OIL¹³ and OWL¹⁴. An example of a medical ontology expressed in OWL can be found in annex 2.

Ontology languages deal with classes or concepts, instances, relations, datatypes and data values. Concepts can often be constructed out of restrictions on a relation or by combining other concepts. Some relations that may be used in ontologies are: specialisation (\leq), exclusion (\perp), instantiation (\in) and assignment ($=$). Models can be made for ontologies: a model M is an interpretation of an ontology O if it satisfies all the assertions in O : $\forall \delta \in O, M \models \delta$ (can be derived from). Many models can exist for one ontology.

The aim of matching ontologies is to reduce heterogeneity. There are different types of heterogeneity requiring different solutions ([12] Ch2.3):

- *Syntactic heterogeneity*
This occurs when two ontologies are not on the same conceptual level, for instance when comparing a directory with a conceptual model, or when different knowledge representation formalisms are used like OWL and F-logic. The solution for this type of heterogeneity lies at the theoretical level, establishing equivalences between languages.
- *Terminological heterogeneity*
This is caused by different names being used for the same concept, because different natural languages were used (French versus English) or different technical sublanguages, or synonyms like Paper, Memo and Article.
- *Conceptual or semantic heterogeneity*
This refers to differences in modeling the same domain of interest. There could be a mismatch in conceptualisation or in the expression of concepts (explicitation). Reasons for these differences may be *differences in coverage* (different domains), *differences in granularity* (different level of detail) or *differences in perspective* (focus on different aspects of same domain).
- *Semiotic or pragmatic heterogeneity*
This is caused by different interpretations of concepts by people. It will be clear that this kind of heterogeneity is difficult to tackle in an automated way.

¹³<http://oil.semanticweb.org/>

¹⁴<http://www.w3.org/TR/owl-features/>

The ontology matching problem is closely related to the heterogeneity issues discussed. Returning to the model theory: matching ontologies amounts to finding an ontology whose *set of models* is included in the *intersection* of the set of models of the two aligned ontologies, being *maximal* for inclusion ([12] Ch2.5). An alignment expresses the correspondences between entities belonging to different ontologies. To get a clear overview of the entities matched we separate them from the ontology language calling them the *entity language*. Relationships between two entities are assigned a degree of confidence, representing the trust in the fact that the correspondence is correct. This can be expressed as a number in the range $[0,1]$, 1 meaning that entities are certainly the same. *Alignment multiplicities*, like *injective* and *surjective*, can be used to qualify relationships in the alignment. *Bijjective* or one-to-one alignments provide the best match and can be reversed.

5.2.3 Matching techniques

Concept- and ontology matching techniques can be classified in different ways. Dimensions that can be used are ([12] Ch3): the input dimension, the process dimension and the output dimension. Matching algorithms operate on different kinds of input: relational input, XML or OWL. Some use only attribute names, others also datatypes and internal structure. They produce different kinds of output: different forms of alignment, different relationships (equivalence, subsumption), different ways to measure correspondence. The main distinctions can be made in the process dimension, however.

Matching techniques can be either name based, structure based, extensional or semantic based. **Name based techniques** compare strings. A difference can be made between *string based methods*, which focus on one string of characters, and *language based methods*, which focus on sets of strings. It will be clear that the fact that the same string is used to label concepts does not mean they are equal. It may be a *homonym*, a word with different meanings, like 'peer' which can mean 'equal' or 'member of nobility'. The fact that different strings are used does not mean concepts are not equal either, the strings may be synonyms like 'villa' and 'mansion'.

The first thing that is usually done in name based techniques is *normalisation*. Irrelevant differences are removed by taking out diacritics, blanks, punctuation and so on. In language based methods this is taken one step further by stopword elimination and by making variations like: 'theory papers', 'theoretical papers' and 'papers on theory'. After this *string equality* is measured, using different methods like the 'Hamming distance' and 'Cosine similarity' which uses vectors ([12] Ch4). Sometimes *path comparison* is used, taking into account the path by which the concept can be derived, for instance wine - red wine - burgundy. Also external resources like lexicons, in which synonyms and homonyms can be found, are used to match names. These are called *extrinsic methods*. This way likely matches between names in ontologies are identified. Name based techniques are useful in case similar strings are used for the same concepts in different ontologies.

In **structure based techniques** either the *internal structure* or the *relational structure* can be used. When comparing the internal structure properties keys, datatypes and domains are compared. Also constraints on properties like multiplicities can be used. Comparing the internal structure is relatively easy to implement in algorithms but it does not provide much useful information when used on it's own. When looking at the relational structure an ontology can be considered a graph with multiple relations. Finding the correspondences can be translated into finding the maximum common directed subgraph ([12] Ch4.3.2).

In **extensional techniques** the focus is on *instances*. When these are available they can

be a good source for matching information independent of the conceptual part. Extensional information is supposed to be less prone to variability since it is linked to entities in the real world. Measures that can be used focus on the frequency with which instances are found in the same classes. A well known measure is the *Jaccard similarity*:

$$\sigma(A, B) = P(A \cap B) / P(A \cup B) \quad (1)$$

$P(X)$ in this formula is the probability of a random instance to be in the set X. This measure is normalised and reaches 1 when $A=B$, that is when two classes from different ontologies share the same set of instances. Other measures are available, like formal concept analysis (FCA) focusing on constraints on properties, and statistical approaches.

In *semantic-based techniques* model-theoretic semantics is used to justify the results ([12] Ch4.5). These are deductive methods. To perform well for a mainly inductive task like ontology matching preprocessing is needed, like providing 'anchors', entities which are declared to be equivalent. Sometimes external ontologies are used as intermediate ontologies to define a common context. Different deductive techniques are used, for instance description logic techniques. Example ([12] Ch4.5.2):

$$smallcompany = company \cap (\leq 5employee)$$

$$SME(smallmediumenterprise) = Firm \cap (\leq 10associate)$$

$$InitialAlignments : Company = Firm; associate \subseteq employee \Rightarrow smallcompany \subseteq SME$$

Explanation: if definitions for *small company* and *SME (small medium enterprise)* are available and it is known how the components of these definitions: company, firm, employee and associate, relate to each other, the relationship between small company and SME can be deduced, in this case the fact that small companies are a subset of SME's.

When properly integrated with inductive techniques these techniques can be very useful in ontology matching.

5.2.4 Matching strategies and matching systems

Matching techniques are the building blocks for matching solutions [12]. Different calculations can be used to compute the similarity between two entities based on results derived from different matching techniques. Most are based on calculating a weighted average over the different input dimensions. Similarities can also be computed on a global level, looking at ontologies as a whole.

A quite different approach is the use of *learning methods* like neural networks or decision trees. These methods have a learning phase in which many correct and incorrect alignments are presented to train the matcher, and a matching phase in which this knowledge is used to match new ontologies. Similar to this is the use of *probabilistic methods* like Bayesian networks, used to model causes and effects. An expert may specify some of the conditional probabilities, after which others can be derived, thus completing the network.

User input is needed in matching solutions, to provide initial alignments, combining matchers and providing feedback to improve matchers. When extracting the final alignment thresholds can be used to prevent getting suboptimal results.

Many matching systems have been presented over the last decade by universities sometimes cooperating with large companies like Microsoft and IBM. These can be divided

in *schema-based systems*, *instance-based systems*, *mixed systems* and *meta-matching systems*. Schema-based systems use schema level information, abstracted away from instances. Instance-based systems use instances, mixed systems use both. Meta-matching systems combine other matching systems rather than providing their own matchers.

The availability of many matching systems increases the need to evaluate them. Few extensive comparisons are available however ([12] Ch7). Evaluations must take all aspects into account: the availability of systems, their capability to provide accurate alignments, their performance and so on. Difficulties that arise are that matching tasks are so different that a system may perform very well on some data and not well at all on other data. Also the choice of evaluation criteria can be quite difficult. The Ontology Alignment Evaluation Initiative¹⁵ provides more insight into the quality of ontology matching.

5.2.5 Processing alignments

After the matching process the resulting alignments can be stored and presented in different formats. Since alignments are often used in a wider context than the matching system itself choosing a format that facilitates combining alignments is important. Language and purpose independence, web compatibility, expressiveness and simplicity are important criteria when choosing a format ([12] Ch8.1.8). Alignments are often only a step towards the ultimate goal of ontology integration and are used as part of a larger *alignment framework*, also containing tools like *ontology editors*.

Alignments derived by ontology matching techniques may not be intuitively acceptable to users. To improve matching techniques using user feedback and to get people to use alignments these alignments will have to be explained. This can be done by explaining the matchers, the matching process, or the logical reasoning behind the results. Arguments in favour of alignments can be met with counter arguments, leading to negotiations between (software) entities.

The final use of ontology matching may be *ontology merging*, obtaining a new, combined ontology, *ontology transformation*, having one ontology as source and the other as target, *data translation*, translating instances from entities of one ontology into instances of connected entities in the other ontology, *mediation*, being used as go-between for two independent software entities, *reasoning*, using the result to create rules to reason with two matched ontologies. The fact that matching techniques are best used in combination, reusing other ontologies, presents a strong case for creating *alignment services*, combining all these aspects in a standardised format.

5.2.6 Conclusions

The pressure on the development of matching strategies has increased strongly by the vast expansion of internet related services. Many initiatives came up, all covering some aspects of this complicated problem. These will have to be integrated and evaluated to come closer to the results desired. The basic problem remains that the exact meaning of concepts is in the head of the person who created it, and we can not program a computer to learn it. Mission impossible? As much or as little as inter human communication ([12] Ch11.3). When people communicate they do not know exactly what the other party means when using a concept. Still, often people succeed in communicating, and these problems never stopped us trying. There is no infallible way to interpret the meaning of concepts, so negotiating and arguing

¹⁵<http://oaei.ontologymatching.org/>

alignments is part of the deal. Much progress can be made in improving the dynamics of matching strategies, bringing it up to speed with human communication.

5.3 Using ontologies to support compliance

Can ontologies be used to support compliance in IT? A number of articles have been published on this subject. One aspect that is covered in these articles is the problem to match compliance ruling and organisational concepts. Many compliance officers say they find it difficult to translate compliance ruling into concrete measures for their organisation [18]. If it were possible to catch compliance ruling in a *pattern* of measures or business rules ontologies could be used to match the generic concepts in regulations with the specific concepts used in an organisation.

At Stanford University extensive research was done into the use of concept matching and ontologies to develop automated compliance assistants [9][22][24]. The need for this was felt because of the vast number of compliance regulations from different sources, federal law, state law, local regulations, sometimes contradicting, often difficult to understand for people outside legal departments [24]. Many regulations are available online, which opens up the possibility to process them in an automated way [22]. They tend to be organized into deep hierarchies which can be preserved using XML [24]. Also, regulations use domain knowledge, making ontologies, in which this knowledge is captured, valuable when processing them [9]. Ontologies are used to match concepts that are found in regulations and to tag relevant parts. Combined with First Order Predicate Calculus logic in which the rules that must be followed to be compliant are captured they can be used to check if compliancy has been achieved [22].

At Purdue university research was done as well into the use of ontologies to achieve compliance [30][31]. This research focuses on privacy ruling. Ontologies are used to match concepts used by different federated organisations and by individuals using the services of these organisations. They are used to establish a common vocabulary, to automatically detect semantic relationships among attributes and to reason about policy subsumption [30]. A federated or reference ontology is used on which all parties agree, and local ontologies are matched to this ontology. The reference ontology is also used to replace concepts with less revealing ones. For instance, to get certain services someone may have to prove to be an adult. This could be done by giving the birthdate in ones passport but also by mentioning the fact that one has a driver license, which could be considered less revealing. Inference rules are used combined with ontologies to protect privacy [31].

Since compliance ruling is domain specific and matching concepts used in different ruling and in organisations is necessary to get insight into what needs to be done to be compliant ontologies could be a useful asset to support compliance.

6 Privacy Compliance Research at Purdue University

6.1 Digital Identity Management and Protection

At the Computer Science department of Purdue University (Indiana, U.S.A.) a research group is dedicated to *Digital Identity Management and Protection*. This research is part of the CERIAS program. CERIAS is the Center for Education and Research in Information Assurance and Security¹⁶. In this center scientists from different disciplines, technical, legal, ethical, economical, linguistic and so on, work together on subjects concerning information security.

Digital identity stands for the information known about an individual in a specific IT environment. It does not just encompass login names (*nyms*) but also identity attributes of users [30]. *Digital identity management* is dedicated to handling digital identities. This involves making sure a person is who he claims to be, preventing people who are not who they claim to be to get access to systems, but also facilitating people to get access when they are entitled to it. Another aspect is preventing the spreading of information about individuals without their consent. Identity theft and information leakage are serious threats now that more and more systems get connected. The research is dedicated to achieving compliance in privacy protection and facilitating digital identity management in information sharing environments. The research group was motivated by corporate initiatives like Liberty Alliance¹⁷ and university initiatives like Shibboleth¹⁸, aimed at facilitating inter organisational collaboration and knowledge sharing [30].

Information sharing environments have become increasingly important with the rise of e-commerce. Many organisations today offer services together with other organisations via the internet. To facilitate people using these services it is necessary to provide them with ways to get access to all service provider environments without having to provide their credentials over and over again. You can compare this to getting a passport from a trusted organisation once, which you can then use as proof of identity all over the world. Protecting user privacy is very important in such environments since users may not want all information about them to be shared.

The CERIAS research concentrates on federated information sharing environments, which consist of groups of organisations working together and building trust among each other to allow sharing of user identity information [4][5]. In these federations Purdue researchers propose a single sign on (SSO), the possibility to access multiple IT systems without having to log in more than once, without dependence on Public Key Infrastructures (PKI)¹⁹, since they feel too many implementation problems are connected to the use of PKI's [5]. Furthermore solutions against identity theft are proposed based on *zero knowledge proof*, using encryption, and distributed hash tables [4]. To prevent single sign ons to become a 'single point of failure' strong authentications have to be provided. A distinction is made between the role of Identity Provider and Service Provider. The role of Identity Provider has to be distributed carefully without losing control. In practice both roles are often combined by participating organisations in a federated environment. A user enlists with a service provider, which functions as an identity provider, and after this the user can use the services of all service providers [5].

User attributes need to be shared because getting access to services increasingly depends

¹⁶<http://www.cerias.purdue.edu/>

¹⁷<http://www.projectliberty.org/>

¹⁸<http://shibboleth.internet2.edu/>

¹⁹binding public keys with user identities by means of an authority providing certificates

on user attributes rather than just on identification. Different approaches are proposed for services that require low clearance and for which uncertified attributes can be used, and for services that require high clearance. A distinction is made between *strong* and *weak* identifiers, the first enabling identification of a user, the second shared by more than one user, making identification impossible (though a combination of weak identifiers might again enable identification). Sensitive attributes are protected by grouping and encrypting them, requesting information on one of them to release information on another to prevent identity theft [4]. The privacy preferences stated by users have to be taken into account in this process. In information sharing environments the need is often felt however to hold users accountable for their actions and to link actions to users at any time, thus limiting anonymity.

Linking users to their actions is also necessary because a user may only have limited access to services [3]. In these cases it is useful to distinguish between service provisioning policies and user authentication policies. Next to these policies we distinguish privacy policies of both users and service providers, and federated policies stated by the federation. An *assertion language* is proposed to collect all data needed for these policies. An assertion or property language can be used to exchange authorisation data and to capture the behavior of users across different cycles and in different places. By querying the information thus assembled about users decisions about granting them access to services can be taken [3].

In open systems like the Internet users' sensitive information needs to be handled even more carefully. *Trust negotiation* is an emerging access control approach for establishing trust in these kind of open environments [5]. A third party is often used as certificate provider and credentials are shared bilaterally among organisations, implying organisations first have to provide sensitive information about a user, showing themselves a trustworthy partner, to get sensitive information. This protocol could also be used for non members within federations. Bilateral credential sharing as such is considered to be useful in negotiations about member access within a federated environment as well [5].

6.2 Traceable and flexible compliance

A number of articles of the Purdue research group are dedicated to achieving *traceable compliance* to privacy ruling when data are exchanged automatically between federated organisations [30][31]. To achieve this means having to comply with different privacy preferences since users may feel differently about sharing their information. Apart from this these preferences might change, so *flexibility* is essential. An automated match has to be made between pre-stated privacy wishes and data requests. For instance, if someone states no medical test results should go to the pharmaceutical industry he does not want to have to state that this concerns both weight and blood test results, blood pressure and X-rays. That means a *common understanding of concepts* between different parties is needed. In the Purdue research this is solved by using a **common ontology**, where concepts are mutually exclusive and form a hierarchy. If someone excludes a higher level concept all underlying concepts are excluded as well. The idea of subsumption is used for this [30]. Concept A subsumes concept B if A is a *specialization* of B, like *apple* and *fruit*. Ontologies are also used to release data that answer a specific request but at the same time reveal as little as possible about the person involved [31].

According to Purdue researchers concept matching using ontologies supports automated and flexible compliance to privacy ruling. It is worthwhile to try and extend this to other types of compliance ruling.

6.3 Purdue proposal for a privacy compliant system

The privacy compliant system described in [30] is set in a medical environment. *Trusted Health* is the name of a federation of medical organisations working together online sharing patients' information. Different organisations store a combination of medical and personal data of patients. To allow proper treatment of patients access to medical data is needed, to ensure for instance payment of treatments personal data, like medical insurance, is shared. These data are stored together with a patients pre-stated privacy preferences.

Researchers at Purdue were contacted during this research to ask for more information and feedback. They referred to other articles of theirs and indicated that the integration of ontologies into their system was one of the most difficult aspects and one of the main subjects of current research.

6.3.1 Matching privacy policies

The system proposed by Purdue researchers consists of different parts. Two versions are proposed, a simple one using privacy preference *templates* (v1) and a more sophisticated one using *customized* privacy preferences (v2). The different parts are: ontologies, privacy preference templates and algorithms.

Ontologies

Simple ontologies, similar to thesauri, are used to enable automated detection of semantic relationships between attributes and to enable reasoning about policy subsumption. If an attribute is protected related attributes should be protected as well. Different words may refer to the same attribute, or attributes can be a specialisation or generalisation of other attributes. Two main classes of attributes are proposed: identity related attributes (patient vital info) and medical attributes (medical record). An example of this can be seen in figure 8.

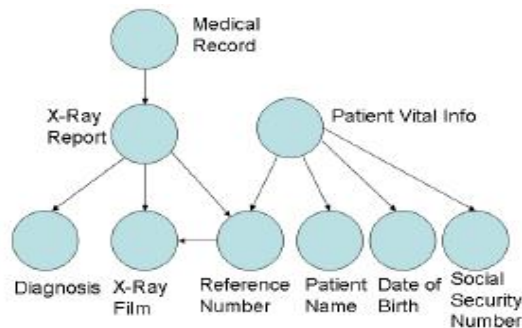


Figure 8: Attribute Graph [30]

An ontology consists of *concepts*, which can be defined as:
A concept C_i is a tuple $[Name_i; KeywordSet_i; DIAttr_i; DomAttr_i]$, for instance $[Xray; (); xray; (xray, medicaldocument, bonesample)]$. In this case Xray is the name of the concept, bonesample and medicaldocument are synonyms. A keyword belongs to exactly one concept,

as does each attribute.

An ontology is a *set of ordered concepts* $[C_1, C_2, \dots, C_n]$. The order relationship \prec represents a generalisation. If $C_i \prec C_k$ then C_k is a generalisation of C_i . In the ontology graph a parent node is a generalisation of a child node, a child node is a specialisation of a parent node (see figure 8). A patient's name, for instance, is a specialisation or a part of a patient's vital info.

It is assumed that the ontology is accepted by all parties involved. In practice ontologies will have to be integrated or concepts will have to be matched using different techniques. More about this can be read in the chapter on concept- and ontology matching.

In proposal v1 limited use is made of ontologies. They are mainly used to relate privacy preference templates to one another. In v2 more extensive use is made of ontologies to relate attributes, purposes, requestors and so on.

Privacy preference templates

To express privacy preferences two different solutions are proposed. First, *privacy preference templates* are used which are in decreasing order of strictness. For this relationship the idea of generalisation can be used as well. If policy T_n is stricter than policy T_{n+1} it can be considered a specialisation of policy T_{n+1} , $T_n \prec T_{n+1}$. If a user wants to use privacy preference template y and a service provider uses template x a simple check if $x < y$ will do to know if $T_x \prec T_y$ and if the privacy rules of the service provider are strict enough to release the requested data.

Preferences are set on the lowest level data. Using three privacy preference templates could look like this:

T1 Strict Policy

Element	Value
<i>Purpose</i>	current
<i>Recipient</i>	ours
<i>Retention</i>	stated-purpose, legal-requirement

T2 Moderate Policy

Element	Value
<i>Purpose</i>	current, analysis
<i>Recipient</i>	ours, same
<i>Retention</i>	business-practice

T3 Casual Policy

Element	Value
<i>Purpose</i>	current, other-purpose
<i>Recipient</i>	ours, other-recipient, unrelated
<i>Retention</i>	indefinitely

The concepts in these policies are taken from EPAL and P3P, languages in which privacy policies can be expressed. In some cases possibilities are added to express generalisation, like: *purpose* current \prec *purpose* current, analysis. In other cases 'wider' terms are used, like: *retention* business-practice \prec *retention* indefinitely.

The second solution uses *customized privacy policies*. For *requested data*, *purpose*, *retention* and *recipient* simple **ontologies** are used. This way different concepts can be related

to one another and different privacy preferences can be set for different sets of data, varying in strictness.

Algorithms

Finally algorithms are proposed to check if privacy policies allow passing on data from one service provider to another. The algorithm used for this can be found in annex 3. First the availability of the requested attribute is checked. Then the privacy preferences of the requestor and sender are compared. If the first is less strict than the latter the request is refused. When privacy preference templates are in order of decreasing strictness, as in version 1, a simple check if the receiving service providers privacy policy is strict enough can be done by checking if the number of his privacy preference template is smaller or equal to that of the sender. This check is done in *IsMoreStrict*.

In the second version, with customized privacy policies, a more complex check needs to be done. The basic algorithm is the same, but the check *IsMoreStrict* is more complicated (annex 4). For every attribute requested it is checked whether all aspects of the privacy policy of the receiver are more strict than those of the sender, otherwise no data are transferred.

6.3.2 Policy tracing

When aiming for compliance there is a choice between trying to prevent breaking of rules at all costs, or leaving some room for people to take their own responsibility. The first approach, trying to prevent breaking of rules at all costs, may sound desirable from a compliance perspective, but it may introduce extreme rigidity in business processes. Also, there has been a shift in compliance from a rule based to a principle based approach, relying more on people's interpretations of regulations [2]. The latter approach requires checking *a posteriori*, making adequate logging of actions necessary [8].

In the Purdue proposal enabling people to check whether their privacy preferences have been respected is an essential part of the system. All requests for data and actions thereupon are logged. A tamper proof logging system is supposed to be in place. If people find their data with a service provider they can follow the line back to check whether all parties involved respected their privacy wishes. If not, the party not being compliant is identified. The tracing algorithm in annex 5 is used for this purpose. First it is checked whether user data is found at the service provider. If so, the line is followed back to the sender, checking privacy preferences to see if they are strict enough to pass on the data. This way the line is followed all the way back to the data owner. If no breach of rules was detected this is reported. If a breach of rules is found the name of the service provider involved is passed on to the user. The algorithm can deal with a single or non-consecutive malicious parties by identifying the first non-compliant party and resuming the tracing after this party. To deal with consecutive malicious parties forced third party checking would be necessary.

7 Using the Ampersand Method to achieve Compliance

7.1 Criteria for a privacy compliant system

Let us look at criteria that can be set for a privacy compliant IT system, derived from the Purdue research into traceable and flexible compliance of privacy policies described in chapter 6. Two important requirements are mentioned, first to provide mechanisms for facilitating privacy policies matching, second to provide mechanisms for users to trace their identity information across a federation. Both criteria are described in more detail below.

7.1.1 Matching privacy policies

To maximize user convenience identity attributes needed to get access to services within the federated environment have to be exchanged. Users' privacy preferences have to be taken into account in this process. A generally accepted vocabulary, like EPAL (Enterprise Privacy Authorization Language)²⁰ and P3P²¹, needs to be used to specify privacy policies to facilitate organisations joining. Both users and service providers specify their privacy policy using templates, or by creating customized policies. *Subsumption* can be used on policies defined over similar classes of data to determine if they conflict or if one implies the other. Privacy preferences state which data can be released to which recipients, for which purpose, and how long the data can be retained.

The following criteria for the privacy compliant system can be set:

1. Facilitate user access and service provider collaboration by allowing sharing of user attributes between service providers.
2. Give users and service providers templates to express their privacy preferences or enable them to create customized privacy policies.
3. Provide a common vocabulary, for instance by using an ontology.
4. Make sure that user attributes are only exchanged if this does not go against the user's privacy preferences.

7.1.2 Policy tracing

Policy tracing is a method to verify that data have been transmitted from FSP_1 (Federated Service Provider) to FSP_k without violating the user's privacy preferences. If not it should be possible to identify the FSP that did not comply with these privacy preferences.

This adds the following criteria for the privacy compliant system to the list:

5. Provide evidence to users that their privacy is protected.
6. Provide a tracing mechanism to users to identify the FSP concerned if their privacy preferences were breached.

²⁰<http://www.zurich.ibm.com/security/enterprise-privacy/epal/>

²¹<http://www.w3.org/TR/P3P/>

7.2 A privacy compliant system using the Ampersand Method

7.2.1 Initial system using privacy preference templates

In the initial system proposed by Purdue researchers participants in the federated network can choose from three privacy preference templates in decreasing order of strictness. The ADL code for a similar system using the Ampersand Method can be found in annex 6.

When running the ADL program using the online compiler²² the relations are analysed and a system is specified or even generated that supports the business logic in the ADL code. The relations can be seen in figure 9. A screenprint with an example of the user interface of the system generated is in figure 10. The user interface is flexible and shows the entities, relationships and rules that are selected.

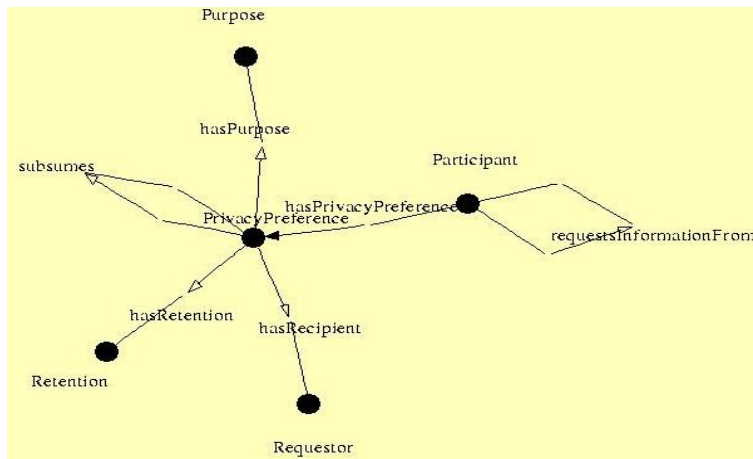


Figure 9: Relations Privacy Preference Templates

In the ADL program first three *privacy preference templates* are defined, covering one or more *purposes*, *retention periods* and *requestors*. The assignments of the latter in this case serve mainly as illustrations since the actual functionality is in the subsumption relation of the templates. This is illustrated by figure 9, only *Participants* and *Privacy Preferences* are part of cycles and are used in business rules.

Then the subsumption of the templates is determined. All subsumption relations are registered, including *grandparent - child* relationships. This facilitates checking whether the privacy policy of the requestor is strict enough (= at least as strict as). After this all participants are connected to a privacy preference template. Finally the information requests are administrated.

One simple business rule is enough to assure that no privacy breaches occur in this setting. It checks, for every request for information, whether the privacy preference of the requestor is equally or more strict than that of the party the data is requested from. If not, a breach of rules is detected and the system is blocked until the situation is corrected, for instance by closing the request that violates the rule.

In this example the rule is broken by the *SP industry* asking data from *MacBeth* and by the *SP hospital* asking data from the *SP university*, since the parties requested to transfer

²²<http://86.88.190.85/ADL.php>

The screenshot shows a software interface with several panels:

- ENTITIES:** A search box containing 'Participant'.
- RELATIONS:** A list of relations including 'hasPurpose', 'hasRecipient', 'hasRetention', 'subsumes', and 'requestsInformationFrom'.
- RULES:** A list containing 'Rule1'.
- PARTICIPANT:** A table with columns 'PARTICIPANT' and 'HASPRIVACYPREFERENCE'.

PARTICIPANT	HASPRIVACYPREFERENCE
Brown	PP1
Fox	PP3
Jones	PP2
MacBeth	PP2
SP1_university	PP1
SP2_hospital	PP2
SP3_industry	PP3
SP4_hospital	PP1
- SUBSUMES:** A table with columns 'PRIVACYPREFERENCE' and 'PRIVACYPREFERENCE'.

PRIVACYPREFERENCE	PRIVACYPREFERENCE
PP2	PP1
PP3	PP1
PP3	PP2
- RULE1:** A section with a 'FIX' button and a table of breaches.

INFORMATION CAN ONLY BE REQUESTED FROM A PARTY WITH AN EQUALLY OR LESS STRICT PRIVACY POLICY.

PARTICIPANT	PARTICIPANT
SP2_hospital	SP1_university
SP3_industry	MacBeth

Figure 10: Screen Privacy Preference Templates

data have stricter privacy policies, see figure 10. These breaches of the rule are reported and need to be resolved. The simplest way to resolve the problems is by closing these requests. Other reactions are possible, more about which can be found further on in this chapter.

7.2.2 System with customized privacy preferences

The ADL code for a more complicated system offering customized privacy preferences can be found in annex 7. Every privacy preference covers one or more purposes, retention periods and requestors. On some aspects a privacy preference can be more strict, on others more lenient than others. Now it is no longer possible to order privacy preferences just like that. They have to be ordered on every dimension: purpose, retention and requestor. To achieve this hierarchies are used, partly based on P3P, the Platform for Privacy Preferences. P3P is a language which can be used to express privacy policies in. It is a standard proposed by W3C, the World Wide Web Consortium. Interaction on the World Wide Web requires common understanding of privacy concepts, and P3P can be used to achieve this in an automated way.

The purpose hierarchy (figure 11) shows for which purposes data can be used. If a party chooses *general purpose* the data can be used for all purposes. Other possible choices are *treatment*, or *research*. It is also possible to choose only *development*, excluding *teaching* and *marketing*, which all subsume *research*. The chosen purposes and the ones subsuming these are allowed according to the privacy preference.

The retention hierarchy (figure 12) determines for how long data can be stored. This hierarchy is a simple one-to-one order from *indefinitely* to *no-retention*.

The recipient hierarchy (figure 13) indicates which parties can receive the data, only the one requesting it (*ours*), also similar parties (*same*), all *known recipients* or *all parties*.

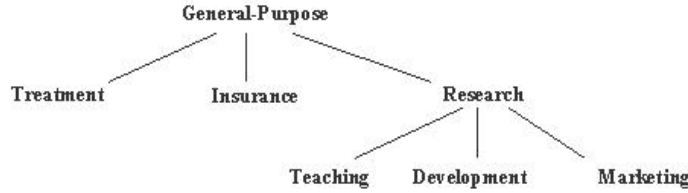


Figure 11: Purpose Hierarchy



Figure 12: Retention Hierarchy

The hierarchies are translated using a subsumption relation, indicating which values subsume which other values. Equal values are also connected, meaning sufficient subsumption is achieved in this context. This way the business rules need not check whether privacy preferences of the requestor are *equally* or *more* strict, a simple check if it is *strict enough* will do. All three hierarchies have their own subsumption relation and can easily be expanded by adding records to the subsumption tables.

Three business rules are used to check if the purpose, recipient and retention of the requestor are equal to or subsumed by those of the holder. Every breach of a rule indicates a dimension in which compliancy was not achieved. Finally in the code different types of participants are distinguished, persons and SP's. The difference is used to make sure only SP's can do a request for information.

Looking at a screen generated with this ADL program (figure 14) three rules are visible, which are all broken by some of the information requests. The request made by the *SP industry* to *MacBeth* breaks all three rules since MacBeth's preferences on all three dimensions are stricter. The request made by the *SP hospital* to the *SP university* only breaks rule 1, since the purpose policy of the university is stricter, the other policies are not. Breaking one rule is enough to 'block' the system however until the breach of rules is amended.

Adding a medical ontology

Taking customization one step further we introduce the possibility to vary privacy preferences based on the *sort of data* concerned. To achieve this a data hierarchy (figure 15), a simple medical ontology, is introduced. The code for this system is also in annex 7.

Every privacy preference is attached to a certain type of data, also covering the data which *subsume* this. This way the use of personal data might be more restricted than the use of medical data and so on. In all three business rules a statement is added that makes sure that the right privacy preferences, referring to the same kind of data, are compared.

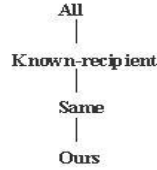


Figure 13: Recipient Hierarchy

The screenshot displays a software interface for customizing privacy preferences. It is divided into several sections:

- ENTITIES:** Lists 'PrivacyPreference' and 'Participant'.
- RELATIONS:** Lists 'hasPurpose', 'hasRequestor', 'hasRetention', 'subsPurpose', 'subsRetention', 'subsRequestor', and 'requestsInformationFrom'.
- RULES:** Lists 'Rule1', 'Rule2', and 'Rule3'.
- PARTICIPANT (HAS PRIVACY PREFERENCE):** A table mapping participants to privacy preferences:

PARTICIPANT	HAS PRIVACY PREFERENCE
Fox	PP1
MacBeth	PP2
SP1_university	PP3
SP2_hospital	PP4
SP3_industry	PP5
- REQUESTSINFORMATIONFROM (H > CSV):** A table mapping requestors to participants:

PARTICIPANT	PARTICIPANT
SP1_university	Fox
SP1_university	SP3_industry
SP2_hospital	MacBeth
SP2_hospital	SP1_university
SP3_industry	MacBeth
- RULE1:**

INFORMATION CAN ONLY BE REQUESTED FROM A PARTY WITH AN EQUALLY OR LESS STRICT PURPOSE POLICY.

PARTICIPANT	PARTICIPANT
SP2_hospital	SP1_university
SP3_industry	MacBeth
- RULE2:**

INFORMATION CAN ONLY BE REQUESTED FROM A PARTY WITH AN EQUALLY OR LESS STRICT REQUESTOR POLICY.

PARTICIPANT	PARTICIPANT
SP3_industry	MacBeth
- RULE3:**

INFORMATION CAN ONLY BE REQUESTED FROM A PARTY WITH AN EQUALLY OR LESS STRICT RETENTION POLICY.

PARTICIPANT	PARTICIPANT
SP3_industry	MacBeth

Figure 14: Screen Customized Privacy Preferences

An example of the invariants created when running this ADL program can be seen in figure 16. This states that if information is requested the *purpose* of the requestor for the data requested has to *subsume* the *purpose* of the participant the request is made to for these data. This invariant is upheld by the system no matter what. Similar invariants are specified for other entities, together covering all business rules.

The *SP hospital* now is allowed to request *MacBeth's medical data* but not his *personal data* since *MacBeth's* privacy preferences concerning personal data are much stricter than those concerning medical data. *MacBeth* allows *no retention* of personal data, whereas the hospital wants to retain the data as needed for the *stated purpose*. This goes against *MacBeth's* retention preference and causes a breach of rule 3.

The *SP industry* is not allowed to see any of *MacBeth's* data, having a more lenient privacy policy for all data sorts than *MacBeth*. Other requests, asking data from parties having at least as strict policies for the data requested, are allowed without any problems.

7.2.3 Building flexibility into compliance

We now have a system that checks compliance with privacy preferences which can be custom made based on the type of data. When a participant in the federated network requests data from another participant in the network this is only allowed if the privacy preferences of

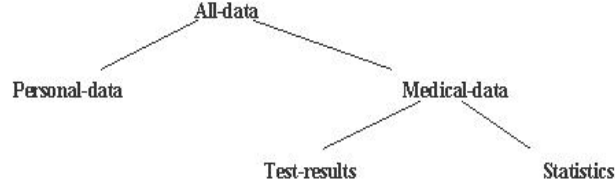


Figure 15: Medical Hierarchy

Invariant: $\forall s :: SP; p :: PrivacyPreference :$
 $s \text{ requestsInformation } p \Rightarrow (\exists p' :: PrivacyPreference; p'', p''' :: Purpose :$
 $s \text{ I belongsTo}(p') \wedge p' \text{ hasPurpose } p'' \wedge p''' \text{ subsPurpose } p'' \wedge p \text{ hasPurpose } p''') \wedge$
 $(\exists p' :: PrivacyPreference : s \text{ I belongsTo}(p') \wedge \text{refersToData}(p) \text{ subsData refersToData}(p'))$

Figure 16: Invariant Purpose

the requestor concerning the requested type of data are equally or more strict than those of the party storing the data. If this is not the case no data can be transferred. This forces compliance but leaves little room for people taking their own responsibility by reacting to a situation following the *intention* of the ruling rather than the rule itself. This kind of flexibility can be achieved quite easily however using Ampersand. In annex 8 the code can be found for a system where data is passed when special permission is given, even if this goes against privacy preferences. To every rule a check is added whether permission was given to transfer data. If so, transferring data is allowed after all.

When comparing the invariants created when running this ADL program (figure 17) with the one in figure 16 a check is added whether permission was given for the data transfer. If so, no breach of the rule occurs.

Invariant: $\forall s :: SP; p :: PrivacyPreference :$
 $s \text{ requestsInformation } p \Rightarrow ((\exists p' :: PrivacyPreference; p'', p''' :: Purpose :$
 $s \text{ I belongsTo}(p') \wedge p' \text{ hasPurpose } p'' \wedge p''' \text{ subsPurpose } p'' \wedge p \text{ hasPurpose } p''') \vee$
 $(\exists p' :: Permission : s \text{ I permissionTo}(p') \wedge \text{permissionConcerns}(p') \text{ I } p)) \wedge$
 $((\exists p' :: PrivacyPreference : s \text{ I belongsTo}(p') \wedge \text{refersToData}(p) \text{ subsData refersToData}(p')) \vee$
 $(\exists p' :: Permission : s \text{ I permissionTo}(p') \wedge \text{permissionConcerns}(p') \text{ I } p))$

Figure 17: Invariant Purpose including Permission

When looking at the relations as analysed by the ADL engine (figure 18) subsumption relations are visible for *purpose*, *retention*, *requestor* and *data sort*. *Permissions* and *information requests* also form a cycle, meaning information requests are only allowed when permission is given concerning this specific request. SP's are the only participants doing requests and getting permission. The other participants, persons, are the ones the information is about, which does not lead to any specific relations in this system.

A datamodel is also created when running an ADL program. The datamodel for this program can be seen in figure 19. Entities, relations and services needed for the system are specified. These specifications can be used to build an IT system implementing the desired functionality.

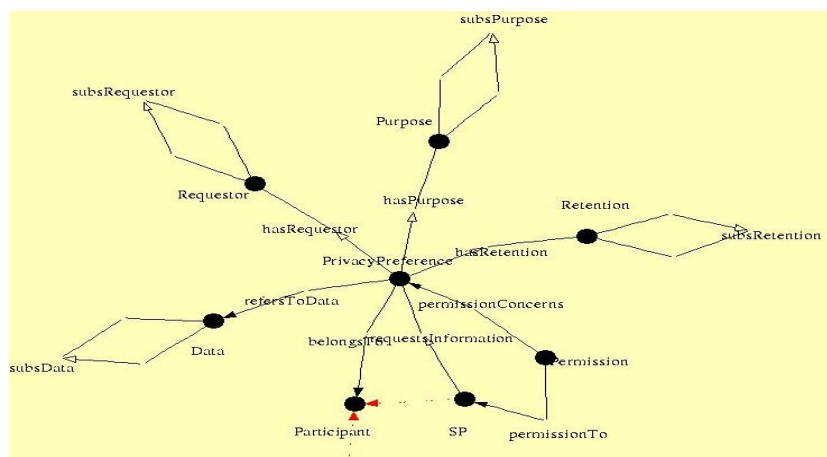


Figure 18: Relations Permission taken into account

Finally an example screen of the IT system that can be generated with the ADL program can be seen in figure 20. The data transfer of medical data from *SP university* to *SP hospital*, for which special permission was given, no longer causes a breach of rules. Transfer of *medical data* from *MacBeth* to *SP industry* and of *personal data* from *MacBeth* to the *SP hospital* are not covered by special permissions and do cause breaches of rules, in the latter case, as mentioned before, only because MacBeth allows no retention of personal data whereas the SP hospital retains these data for the stated purpose.

Aligning compliance and business policy

Organisations have to decide for themselves how to deal with these situations, for instance by asking data owners for their permission in special cases, or by asking higher management to sign off. These permissions have to be logged to allow checking of compliance and to be able to make people accountable for their actions. This way more flexible compliance can be obtained. Business policy can be captured in the business rules and will be upheld by the system.

7.2.4 Checking loggings

The Purdue proposal covers one more aspect. It allows checking of loggings to detect breaches of compliance with privacy ruling. The ADL for a similar proposal made using the Ampersand Method can be found in annex 9. The basis of the system, consisting of the determination of privacy preferences, remains the same. Instead of processing information requests loggings are checked, which state when which data was transferred from which holder to which requestor. The rules for checking compliance largely remain the same as well, except that they are applied on the logging table.

In annex 10 all rules that are specified when running the ADL program can be found. These rules reflect the business rules specified (rules 1-3) but also properties of relations (multiplicities). For instance the properties Transitive and Asymmetric, given to the subsumption relations, can be found in rules 7-10 where these relations are qualified as 'ordeningsrelaties'. These rules, again partly in Dutch due to the engine used, can be discussed with future users

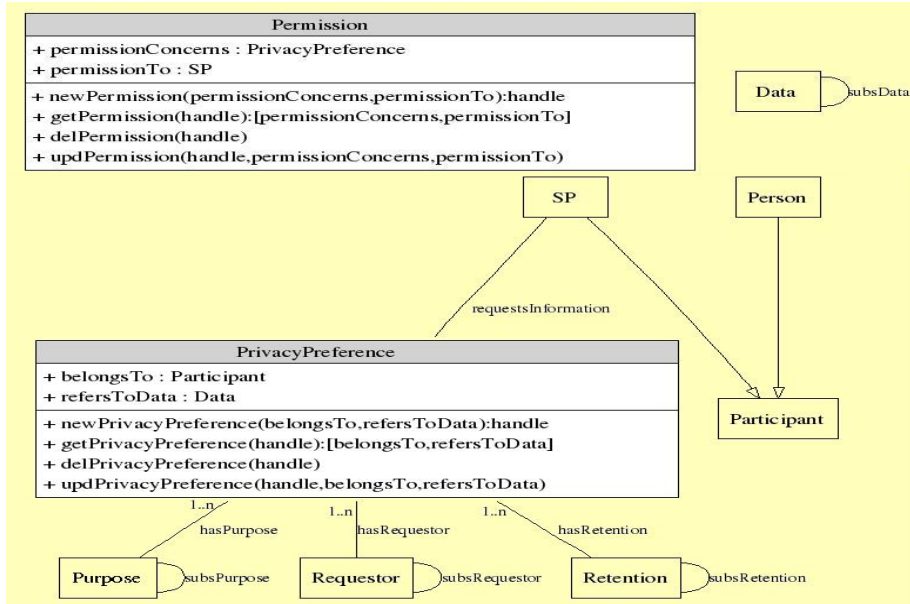


Figure 19: Datamodel Permission taken into account

of the system to make sure they are correct and complete. If not, now is the time to change them to avoid higher costs when having to correct errors after the system has been built.

An example screen of the IT system that can be generated with the ADL program can be seen in figure 21. Please check the examples in the code to understand why only one data transfer, that of *personal data* from *MacBeth* to the *SP hospital*, breaks the rules. For the other logged entries either no privacy preferences were violated or special permission was given to transfer data.

In practice the logging table will have to be stored on a specific part of the network, collecting loggings of every data transfer. The Purdue researcher suppose a *tamper free* logging system for this purpose.

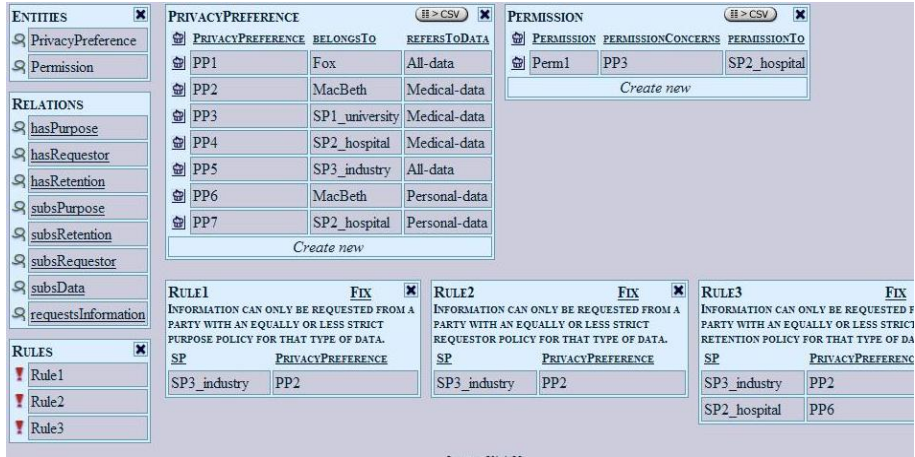


Figure 20: Screen Permission taken into account

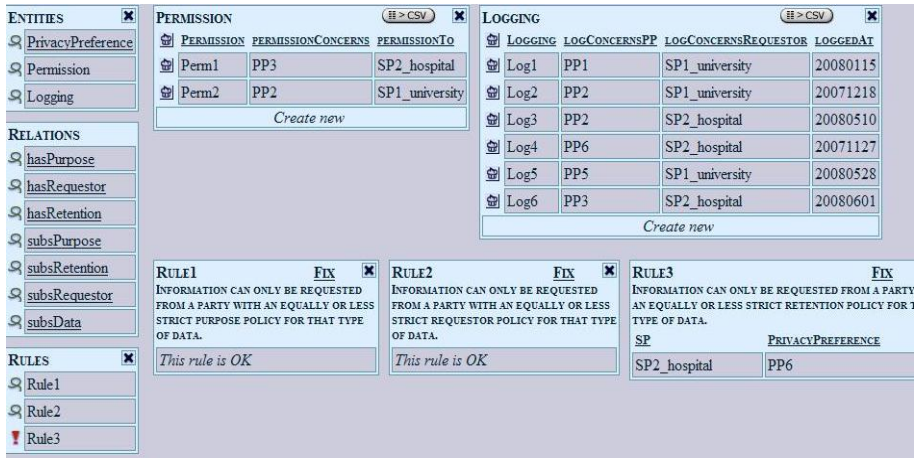


Figure 21: Screen Checking Loggings

8 Comparison Purdue and Ampersand proposals

How do the Purdue and Ampersand proposals compare? Let us go back to the criteria set for the privacy compliant system:

1. Facilitate user access and service provider collaboration by allowing sharing of user attributes between service providers.
2. Give users and service providers templates to express their privacy preferences or enable them to create customized privacy policies.
3. Provide a common vocabulary, for instance by using an ontology.
4. Make sure that user attributes are only exchanged if this does not go against the user's privacy preferences.
5. Provide evidence to users that their privacy is protected.
6. Provide a tracing mechanism to users to identify the FSP concerned if their privacy preferences were breached.

Both proposals facilitate user access and service provider collaboration by allowing sharing of user attributes and providing templates or custom made facilities to express privacy preferences. The services offered in these areas are relatively similar, although the Purdue proposal also checks availability of data. This can be done using the Ampersand Method as well, but since this research is focused more on flexible compliance this was not elaborated upon.

Both proposals provide simple ontologies which can easily be extended. When it comes to making sure that user attributes are only exchanged if this does not go against a user's privacy preferences and providing evidence of this to users the Ampersand Method offers some clear advantages. All logic used to generate the system is in the business rules, which can be found in one place, making it easier for users to check. Mathematical proof can be given that the system operates according to these specifications [19]. In the Ampersand proposal business rules are used directly to specify and generate the system. These are the invariants of the system which will be upheld at all times. In the Purdue approach business processes need to be programmed to uphold the business rules. This leaves room for misinterpretation of the way rules should be implemented.

The Ampersand Method is based on relations and ontologies, integrating business ontologies into ADL aligns with the basic strategy of this language. In the Purdue proposal integrating ontologies requires extensive programming. Purdue researchers indicated that this was one of the more complex issues in their proposal (mail Anna Squicciarini May 2008).

When it comes to flexibility in the Ampersand proposal the desired business policy only needs to be translated into the business rules to get the desired functionality, available for users to check. In the Purdue proposal the desired business processes, which are the **result** of business policy, need to be formulated. The logic to enforce the desired functionality is located in coding implementing these business processes, making it difficult for users to check.

A posteriori checking can be done both in the Purdue and Ampersand proposal. In the Ampersand proposal all breaches of rules are detected at once. The Purdue tracing system follows the trace back to the first participant breaching the rules, and picks up the tracing after this. It depends on the environment and the (im)possibilities this offers what the best solution is.

Overall both proposals offer similar functionality to support automated data exchange in

federated environments, complying with privacy preferences. The Ampersand Method offers more easily verifiable compliance to privacy ruling, simple integration of business ontologies, and above all the advantage of being able to use compliance rules directly to generate a system which will uphold these rules, instead of having to program the business processes to uphold the rules leaving room for misinterpretations.

9 Conclusions and Further Research

9.1 Ampersand, ontology matching and compliance

In this research two proposals were presented for an IT environment in which federated parties can exchange data in an automated way, complying with pre-stated privacy wishes. One proposal was made by researchers at Purdue university. The other proposal was made using the Ampersand Method for systems development.

Let us return to the research questions posed in this thesis:

1. Is it possible to create an IT environment (or functional specifications for this) using the Ampersand Method, which provides traceable and flexible compliance with privacy policies, as described in the article by researchers of Purdue University [30]?
2. How does this environment compare to the one proposed by the researchers at Purdue and what does this tell us about the usability of the Ampersand Method to achieve compliance?
3. Can the use of business ontologies facilitate achieving compliance when using the Ampersand Method?

Starting with the first question this research shows that creating an IT environment for automated information exchange using the Ampersand method is feasible in such a way that flexible and traceable compliance with privacy preferences is achieved. It also confirms the research done at Purdue University by showing that creating a federated environment for automated information exchange can be useful and need not compromise the privacy of people involved.

Coming to the second question this research shows that when it comes to **proving compliance** the Ampersand Method offers clear advantages. All logic used to generate the system is in the business rules, which can be found in one place, making it easier for users to check. Mathematical proof can be given that the system operates according to these specifications [19]. Also, in the Ampersand proposal business rules are used **directly** to specify and generate the system. These are the invariants of the system which are upheld at all times. In the Purdue approach business processes need to be programmed to uphold business rules. This leaves room for misinterpretation of the way rules should be implemented and implementation errors during systems development.

Proving correct implementation of ruling is especially interesting in compliance, but not only in that area. All organisations using information systems want the desired functionality to be implemented and want to be able to 'verify' this. Using the Ampersand Method enables businesses to check whether the rules implemented represent their business logic, and this supports IT systems development in other areas as well.

Finally, addressing *the third question*, this research confirms the fact that using business ontologies can be beneficial when trying to achieve compliance using the Ampersand Method. Business ontologies provide a common set of concepts and enable matching of concepts, and their use matches the ADL strategy. In compliance, ontologies can bridge the gap between concepts used in compliance ruling and in organisations. Their use can be extended outside compliance to all situations where parties use different concepts.

In the case presented in this research ontologies containing privacy policy concepts from EPAL and P3P were used, and ontologies containing medical data. These ontologies contain the relevant business concepts in this example case. When concerned with for instance

waste control in the chemical industry relevant business concepts could be found in ontologies containing chemical and waste control concepts. By choosing ontologies which contain the relevant business concepts concept matching and ontologies can be used to bridge the gap between concepts in generic rules and concepts within an organisation. In dynamic environments like the internet ontology matching is a very complicated task. In more static environments automated and manual matching can be combined to achieve good quality alignments.

As an added benefit this research showed that with a limited number of business rules substantial functionality can be generated. Acquiring the skills needed to formulate business rules in ADL is not complicated compared to programming in most functional programming languages offering similar advantages concerning delivery of mathematical proof. Most of the complicated work is done behind the screens, in the Ampersand engine. Productivity of IT developers who manage to master these skills can increase substantially when using the Ampersand Method. This way more home made customisation could be achieved. It would be too optimistic to assume that all IT staff could take this step however.

9.2 Further research

Combining the Ampersand Method with the use of business ontologies shows promising results when trying to achieve compliance. Business ontologies and concept matching can be used to integrate predefined compliance rules with an organisation's own business rules. Further research is required to show the feasibility of matching these *compliance patterns* to existing business rules in an automated or semi automated way using business ontologies. Given the fact that compliance ruling is abundant and often changes, and the fact that most compliance officers find it very difficult to translate compliance ruling into workable measures for their organisation, supporting this would be a major improvement. Research into combining *rule patterns* using (business) ontologies is useful in many areas of IT systems development, not only to achieve compliance. It could also be used to combine subsystems into a system offering enhanced functionality.

Generating a *compliance certificate* with the correct implementation of a *compliance pattern* could be a very interesting bonus for organisations struggling to show they did what needed to be done to get compliant. Research needs to be done into the feasibility of creating compliance patterns and proving correct matching of these patterns with existing business rules. The Ampersand Method covers proving the correct implementation of business logic.

Finally, very practical research could be done into the best way to integrate business ontologies in different formats with the ADL compiler, facilitating the combined use of both.

Reflections on Research

During this research I became very enthusiastic about the simple yet stunningly effective features of the Ampersand Method. Catching *business logic* in logically sound rules and using these to derive business processes and even generate functionality is very effective to make sure users get the functionality they asked for. Confronting users with the consequences of what they asked for using the Ampersand Method helps them realise what they really want and supports getting complete and consistent specifications. The Ampersand entities *concepts*, *relations* and *rules* can be discussed with users, especially when translated into 'natural language' by the Ampersand engine. The moment I realised that, by using business rules to define all *allowed situations* and checking if the state of a system is still within its allowed boundaries after an action, a system can be generated that upholds business logic, was wonderful. As is often the case with logical insights I could not imagine why it had not occurred to me before! Though I do not want to present the Ampersand Method as a Panacea, for programming sequences of business actions for instance I feel a different approach might be easier to use, this Ampersand feature alone justifies further research. Having a firm *business background*, currently operating as part of the business rather than IT, I know how many problems occur because desired functionality is misunderstood and specifications are wrongly implemented. It would make life so much easier, not to mention save a lot of money and frustrations, if we could close (part of) the gap between business and IT.

This research applies the Ampersand Method in the area of compliance and IT systems development and builds upon claims of this method. The consequence of this is that *knowledge* of the Ampersand Method is needed to fully appreciate the achievements. I tried to provide some basic knowledge realising that more is needed to get the full picture. I hope the references will fill the gap for those wanting to know more.

When looking beyond the Ampersand Method this research shows how business rule driven development and the use of business ontologies offer advantages in the area of compliance and IT, but also in other areas of IT systems development. The interest in concept- and ontology matching increased due to their usability on the internet and much research has been done in this area over the past years. The results of this research can be used in other areas than compliance as well. This thesis provides more theoretical background on ontology matching than needed. This knowledge may help form ideas about further research however, to enhance possibilities in the research area described.

Returning to the Ampersand Method, this research confirms that this method offers some interesting advantages in the area of IT systems development. To many organisations it will be yet another method of IT systems development however, for which they would have to find people to support it. It will take strong evidence of advantages in terms of costs or enhanced functionality to convince these organisations to give it a try. Software developers looking for a way to develop standard software on a larger scale may sooner see the benefits. The main ideas described in this thesis can be used in other environments as well however, hopefully making this research interesting to more parties. The strong points of the Ampersand Method I feel justify further research and I am sure (part of) this approach will find its way in the IT landscape.

All sites were accessed on September 6th 2008.

References

- [1] Andersson T et al. *Key success factors for a sustainable compliance with section 404 of the Sarbanes-Oxley act*. Master thesis Business Administration. Goeteborg University Sweden. 2006.
<http://www.handels.gu.se/epc/archive/00004925/01/05-06-106M.pdf>
- [2] Barends R. *Activeren van de administratieve organisatie*. Master thesis Computer Science. Open University of the Netherlands. 2003.
- [3] Bertino E et al. *Policy Languages for Digital Identity Management in Federation Systems*. Seventh IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'06). 2006.
<http://homes.cerias.purdue.edu/~bhargav/IdM/polrev3.pdf>
- [4] Bhargav-Spantzel A et al. *Establishing and Protecting Digital Identity in Federation Systems*. Journal of Computer Security (2006) volume 14 number 3.
<http://homes.cerias.purdue.edu/~bhargav/IdM/jcssubmitColor.pdf>
- [5] Bhargav-Spantzel A et al. *Integrating Federated Digital Identity Management and Trust Negotiation*. IEEE Security and Privacy Magazine (2005) volume 46.
<https://www.cerias.purdue.edu/assets/pdf/bibtex-archive/2005-46.pdf>
- [6] Brink C et al. *Relational Methods in Computer Science*. Springer-Verlag Wien New York. 1996.
- [7] Castano S et al. *H-Match: an Algorithm for Dynamically Matching Ontologies in Peer-based Systems*. Proceedings of the 1st VLDB Int. Workshop on Semantic Web and Databases. Berlin Germany. 2003.
<http://www.cs.uic.edu/~ifc/SWDB/papers/Castano-et-al.pdf>
- [8] Cederquist J et al. *Audit-based compliance control*. International Journal of Information Security (2007) volume 6 issue 2.
<http://www.springerlink.com/content/0q026820j3t1n1kv/fulltext.pdf>
- [9] Cheng C P et al. *Mapping Regulations to Industry Specific Taxonomies*. Proceedings of the 11th international conference on Artificial intelligence and law. Stanford USA. 2007.
<http://portal.acm.org/citation.cfm?id=1276318.1276329>
- [10] Dijkman R et al. *Calculating with Concepts: a Technique for the Development of Business Process Support*. Workshop pUML group. Canada 2001.
<http://portal.acm.org/citation.cfm?id=647247.719643>
- [11] Dijkman R. *Calculating with Concepts, a Formal Conceptual Modelling Technique applied in the Field of Process Architecture*. Master thesis Computer Science. University of Twente Netherlands. 2001.

- [12] Euzenat J, Shvaiko P. *Ontology Matching*. Springer Verlag. ISBN 978-3-540-49611-3. 2007.
<http://book.ontologymatching.org/>
- [13] Fuentes J et al. *Errors in the UML Metamodel?*. ACM SIGSOFT Software Engineering Notes (2003) volume 28 issue 6.
<http://portal.acm.org/citation.cfm?doid=966221.966236>
- [14] GARP (Global Association of Risk Professionals). *An introduction to Risk and Regulation in Banks*. ISBN 1-933861-00-2. 2006.
- [15] Green P, Rosemann M. *Integrated Process Modeling: an Ontological Evaluation*. Information Systems (2000) volume 25 issue 2.
- [16] Gruber T. *Toward Principles for the Design of Ontologies Used for Knowledge Sharing*. International Journal Human-Computer Studies (1995) volume 43 issues 5-6.
<http://tomgruber.org/writing/onto-design.pdf>
- [17] Hinson G. *The state of IT auditing in 2007*. The EDP Audit, Control and Security Newsletter (2007) volume 36 issue 1.
<http://www.informaworld.com/smpp/content?content=10.1080/07366980701547065>
- [18] Hoogenboom B. et al. *Compliance Survey 2007*. NIVRA-Nyenrode Press Netherlands. 2007.
- [19] Joosten S. *Rule based design*. (book, under construction)
<http://icommas.ou.nl/WikiOwi/images/9/99/ADL-draft-2007nov.pdf>
part of this text is available in:
Joosten S, Aarts C, Van Oosten D. *Rule Based Design*, module 'Business Rules' T.18.3.1.1, Open University of the Netherlands. Heerlen. 2008
- [20] Joosten S. *Deriving Functional Specifications from Business Requirements with Ampersand*. (article, under construction)
- [21] Joosten S, Joosten R. *Will Rule Based BPM obliterate Process Models?*. Wiki Open University Netherlands. 2007.
<http://icommas.ou.nl/WikiOwi/images/a/a2/RulebasedBPM.pdf>
- [22] Kerrigan S, Law K. *Logic-Based Regulation Compliance-Assistance*. Stanford University USA. 2003.
<http://eil.stanford.edu/publications/shawn-kerrigan/ICAIL2003-kerrigan.pdf>
- [23] Landelijk Architectuur Congres. *Compliance als strategisch wapen in het Enterprise Architectuur Domein*. 2005.
- [24] Lau G et al. *An E-Government Information Architecture for Regulation Analysis and Compliance Assistance*. ACM International Conference Proceedings series (2004) volume 60.
<http://portal.acm.org/citation.cfm?id=1052279>

- [25] Mercury (whitepaper). *Sustainable compliance, industry regulation and the role of IT-governance*. 2005.
<http://www.mercury.com>
- [26] Naumenko A, Wegmann A. *A Metamodel for the Unified Modeling Language*. Springer-Verlag Berlin Heidelberg. 2002.
<http://www.springerlink.com/content/7g87thcfhb8nm854/>
- [27] Naumenko A, Wegmann A. *Proposal for a formal foundation of RM-ODP concepts*. Swiss federal institute of technology. Lausanne. 2001.
<http://alloy.mit.edu/contributions/NaumenkoWGF01.pdf>
- [28] Noy N, McGuinness D. *Ontology Development 101: a Guide to creating your first Ontology*. Stanford University USA. 2001.
<http://protege.stanford.edu/publications/ontology-development-ontology101.pdf>
- [29] Object Management Group. *Unified Modeling Language Specification*, version 2.1.2. 2008.
<http://www.omg.org/technology/documents/formal/uml.htm>
- [30] Squicciarini A. et al. *Traceable and Automatic Compliance of Privacy Policies in Federated Digital Identity Management*. 6th Workshop on Privacy Enhancing Technologies. Cambridge University UK. 2006.
<http://www.petworkshop.org/2006/preproc/preproc05.pdf>
- [31] Squicciarini A. et al. *Achieving Privacy in Trust Negotiations with an Ontology-Based Approach*. IEEE Transactions on Dependable and Secure Computing (2006) volume 3 issue 1.
<http://doi.ieeecomputersociety.org/10.1109/TDSC.2006.3>
- [32] Wyssusek B. *Ontology and Ontologies in Information Systems Analysis and Design: a Critique*. Journal of the American Society for Information Science and Technology (2007) volume 58 issue 6.
<http://user.cs.tu-berlin.de/wyssusek/publications/Wyssusek-2004-Ontology-and-ontologies-in-ISAD-a-critique.pdf>

Appendices

Appendix 1 - ADL *Bankaccounts and transactions*

Example ADL program created for this research.

CONTEXT Bank

CONCEPT "Client" "Person who uses the services of a bank." ""

CONCEPT "Account" "Account of client in a bank." ""

CONCEPT "Transaction" "Order by a client to transfer money from an account." ""

PATTERN Transactions

belongsTo :: Account → Client [INJ,SUR]

PRAGMA "" " belongs to "

= [("A1001", "Parker")

; ("A1002", "Jones")

; ("A1003", "Stevenson")

; ("A1003", "Brown") – violates FUN

; ("A1004", "Jones") – violates INJ

].

isDoneBy :: Transaction → Client

PRAGMA "" " is ordered by "

= [("T01", "Jones")

; ("T02", "Parker")

; ("T03", "Parker") – violates rule 1

].

takesMoneyFrom :: Transaction → Account

PRAGMA "" " takes money from "

= [("T01", "A1002")

; ("T02", "A1001")

; ("T03", "A1003")

].

– RULES –

isDoneBy~; takesMoneyFrom -: belongsTo~

EXPLANATION "A client can only order a transaction that takes money from his own account."

ENDPATTERN

ENDCONTEXT

Appendix 2 - Example medical ontology (OpenGALEN)

```
<?xmlversion = "1.0" encoding = "UTF - 8"? >
< rdf : RDFxml : base = "http : //www.opengalen.org/open/crm/crm - anatomy.owl"
xmlns = "http : //www.opengalen.org/open/crm/crm - anatomy.owl"
xmlns : owl = "http : //www.w3.org/2002/07/owl"
xmlns : rdf = "http : //www.w3.org/1999/02/22 - rdf - syntax - ns"
xmlns : rdfs = "http : //www.w3.org/2000/01/rdf - schema" >
< owl : Ontologyrdf : about = "" >
< rdfs : comment > OpenGALENontologyforHumanAnatomy < /rdfs : comment >
< /owl : Ontology >
< owl : Classrdf : ID = "MalignantNeoplasticLesion" >
< rdfs : subclassOf >
< owl : Classrdf : ID = "Pathology" / >
< /rdfs : subclassOf >
< /owl : Class >
< owl : Classrdf : ID = "OpenGALEN - Substance - Metaclass" >
< rdfs : subclassOf >
< owl : Classrdf : ID = "OpenGALEN - Ontology" / >
< /rdfs : subclassOf >
< /owl : Class >
< owl : Classrdf : ID = "PostsynapticCell" >
< rdfs : subclassOf >
< owl : Classrdf : ID = "Cell" / >
< /rdfs : subclassOf >
< /owl : Class >
< owl : Classrdf : ID = " - -ControlOfFlowwhichhasEffectiveness
- -EffectivenesswhichhasAbsoluteStateeffective - - - -" >
< rdfs : label > - - ControlOfFlowwhichlt;hasEffectiveness
- -Effectivenesswhichlt;hasAbsoluteStateeffective
gt; - - gt; - - < /rdfs : label >
< rdfs : subclassOf >
< owl : Classrdf : ID = "Process" / >
< /rdfs : subclassOf >
< /owl : Class >
< owl : Classrdf : ID = "MajorBodyDivision" >
< rdfs : subclassOf >
< owl : Classrdf : ID = "OpenGALENAnatomyMetaclass" / >
< /rdfs : subclassOf >
< /owl : Class >
< owl : Classrdf : ID = "SystemMetaclass" >
...
< /AnatomySubpart >
< /rdf : RDF >
```


Appendix 3 - Purdue code *Checking privacy preferences*

Algorithm 1 FSP1 services FSP2's request

Require: *Request*

```
1: Attr  $\leftarrow$  Request.Attribute
2: userID  $\leftarrow$  Request.userID
3: Policy  $\leftarrow$  Request.getPolicyOf(Attr)
4: myPolicy  $\leftarrow$  this.userID.getPolicyOf(Attr)
5: if Attr  $\notin$  this.userID.AttrList then
6:   this.log.Add(Request, notFound, time)
7:   return notFound
8: end if
9: if isMoreStrict(Policy, myPolicy) then
10:  this.log.Add(Request, tooStrict, time)
11:  return notFound
12: end if
13: this.log.Add(Request, released, time)
14: return this.userID.getAttribute(Attr)
```


Appendix 4 - Purdue code *Checking customized privacy preferences*

```
Algorithm 2 isMoreStrict(Pol1, Pol2)
Require: Pol1, Pol2 are objects
1: //For all data elements of Pol1
2: for all E1 | E1 ∈ Pol1.dataElements do
3:   //Get corresponding element from Pol2
4:   E2 ← getElement(Pol2, E1.name);
5:   if E2 == NULL then
6:     return NO
7:   end if

8:   //For all purposes of Pol1.E1
9:   for all P1 | P1 ∈ Pol1.getPurps(E1.name) do
10:    P2 ← getPurp(Pol2, E2.name, P1.name);
11:    if P2 == NULL || P1 ⊆ P2 then
12:      return NO
13:    end if
14:   end for

15:   //For all retentions of Pol1.E1
16:   for all Ret1 | Ret1 ∈ Pol1.getRets(E1.name) do
17:    Ret2 ← getRets(Pol2, E2.name, Ret1.name);
18:    if P2 == NULL || Ret1 ⊆ Ret2 then
19:      return NO
20:    end if
21:   end for

22:   //For all recipients Pol1.E1
23:   for all Rec1 | Rec1 ∈ Pol1.getRecs(E1.name) do
24:    Rec2 ← getRets(Pol2, E2.name, Rec1.name);
25:    if Rec2 == NULL || Rec1 ⊆ Rec2 then
26:      return NO
27:    end if
28:   end for
29: end for
30: return YES
```


Appendix 5 - Purdue code *Check logging privacy compliance*

Algorithm 3 Policy Tracing Message Handlers

Require: TargetFSP T, DataObject D {sender, receiver, owner }

```
1: GM_START:
2: PostMessage(GM_START, inflow)
3: if (T.containsData(D)) then
4:   PassMessage(GM_TRACE (T,D,inflow), D.sender)
5: else
6:   PostMessage(GM_FAILURE)
7: end if
8: GM_TRACE(Target T, DataObject D, flowtype f):
9: if (f == inflow and I am D.owner) then
10:   PostMessage(GM_SUCCESS)
11: else
12:   if (f==inflow) then
13:     sender = D.sender
14:     if (localMatchVerify(sender, my ID, data) == TRUE) then
15:       PassMessage(GM_TRACE(T,D,f),sender)
16:     else
17:       PassMessage(GM_TRACEFAIL(T,D,my ID, outflow)
18:     end if
19:   end if
20:   if (f==outflow) then
21:     receiver = D.receiver
22:     if (localMatch(my ID, receiver, data) == TRUE) then
23:       PassMessage(GM_TRACE(T,D,f),receiver)
24:     else
25:       PassMessage(GM_TRACEFAIL(T,D,my ID, inflow)
26:     end if
27:   end if
28: end if

29: GM_TRACEFAIL(Target T, DataObject D, FailurePoint P, flowtype f):
30: if (f == inflow and I am D.owner) or (f == outflow and I am T ) then
31:   signal FALSE
32:   PostMessage(GM_FAILURE || at point P)
33: else
34:   if (f==inflow) then
35:     sender = D.sender
36:     PassMessage(GM_TRACEFAIL(T,D,P,f)
37:   end if
38:   if (f==outflow) then
39:     receiver = D.receiver
40:     PassMessage(GM_TRACEFAIL(T,D,P,f)
41:   end if
42: end if
```


Appendix 6 - ADL *Privacy preferences using templates*

ADL program created for this research.

CONTEXT PrivacyCompliance

CONCEPT "Participant" "party in federated network, person or service provider." ""
CONCEPT "PrivacyPreference" "a policy statement how to deal with information." ""
CONCEPT "Purpose" "the purpose for which information may be used." ""
CONCEPT "Requestor" "the type of organisation which requests information." ""
CONCEPT "Retention" "the time during which information may be kept." ""

PATTERN InformationRequest

– defining privacy templates

hasPurpose :: PrivacyPreference * Purpose [TOT]

PRAGMA "" " states that information can be used for purpose "

```
= [ ("PP1", "current")  
; ("PP2", "current")  
; ("PP2", "analysis")  
; ("PP3", "current")  
; ("PP3", "other")  
].
```

hasRecipient :: PrivacyPreference * Requestor [TOT]

PRAGMA "" " states that information can be transferred to requestor "

```
= [ ("PP1", "ours")  
; ("PP2", "ours")  
; ("PP2", "same")  
; ("PP3", "ours")  
; ("PP3", "other")  
].
```

hasRetention :: PrivacyPreference * Retention [TOT]

PRAGMA "" " states that information can be kept as long as needed for "

```
= [ ("PP1", "stated-purpose")  
; ("PP2", "legal-requirement")  
; ("PP2", "stated-purpose")  
; ("PP3", "indefinitely")  
].
```

– subsumption of privacy templates

subsumes :: PrivacyPreference * PrivacyPreference [TRN,ASY]

PRAGMA "" " subsumes, is less strict than "

```
= [ ("PP3", "PP2")  
; ("PP2", "PP1")  
; ("PP3", "PP1")  
].
```

– choice of privacy templates
hasPrivacyPreference :: Participant → PrivacyPreference
PRAGMA "" " conforms to privacy preference "
= [("Brown", "PP1")
; ("Jones", "PP2")
; ("Fox", "PP3")
; ("MacBeth", "PP2")
; ("SP1-university", "PP1")
; ("SP2-hospital", "PP2")
; ("SP3-industry", "PP3")
; ("SP4-hospital", "PP1")
].

– requesting information
requestsInformationFrom :: Participant * Participant
PRAGMA "" " requests information from "
= [("SP1-university", "Fox")
; ("SP2-hospital", "Jones")
; ("SP3-industry", "MacBeth") – violates rule
; ("SP4-hospital", "MacBeth")
; ("SP1-university", "SP3-industry")
; ("SP2-hospital", "SP1-university") – violates rule
; ("SP4-hospital", "SP1-university")
; ("SP4-hospital", "SP2-hospital")
].

– RULES –
requestsInformationFrom -: (hasPrivacyPreference; hasPrivacyPreference ~) \∨ (hasPrivacyPreference; subsumes ~; hasPrivacyPreference ~)
EXPLANATION "Information can only be requested from a party with an equally or less strict privacy policy."

ENDPATTERN
ENDCONTEXT

Appendix 7 - ADL *Customized privacy preferences*

ADL program created for this research.

CONTEXT PrivacyCompliance

```
CONCEPT "Participant" "party in federated network, person or service provider." ""
CONCEPT "Person" "person using services of federated network." ""
CONCEPT "SP" "party providing services in federated network." ""
CONCEPT "PrivacyPreference" "a policy statement how to deal with information." ""
CONCEPT "Purpose" "the purpose for which information may be used." ""
CONCEPT "Requestor" "the type of organisation which requests information." ""
CONCEPT "Retention" "the time during which information may be kept." ""
CONCEPT "Data" "the type of data that can be stored of a person." ""
```

PATTERN InformationRequest

– introducing sub entities

GEN Person ISA Participant

GEN SP ISA Participant

– defining privacy preferences

hasPurpose :: PrivacyPreference * Purpose [TOT]

PRAGMA "" " states that information can be used for purpose "

```
= [ ("PP1", "General-purpose")
; ("PP2", "Treatment")
; ("PP2", "Insurance")
; ("PP2", "Development")
; ("PP2", "Teaching")
; ("PP3", "Teaching")
; ("PP3", "Development")
; ("PP4", "Treatment")
; ("PP4", "Insurance")
; ("PP5", "Research")
; ("PP6", "Treatment")
; ("PP7", "Insurance")
; ("PP7", "Treatment")
].
```

hasRequestor :: PrivacyPreference * Requestor [TOT]

PRAGMA "" " states that information can be transferred to requestor "

```
= [ ("PP1", "All")
; ("PP2", "Known-recipient")
; ("PP3", "Same")
; ("PP4", "Ours")
; ("PP5", "All")
; ("PP6", "Ours")
; ("PP7", "Same")
```

].

```
hasRetention :: PrivacyPreference * Retention [TOT]
PRAGMA "" "" states that information can be kept as long as needed for ""
= [ ("PP1", "Indefinitely")
; ("PP2", "Legal-requirement")
; ("PP3", "Business-practice")
; ("PP4", "Stated-purpose")
; ("PP5", "Indefinitely")
; ("PP6", "No-retention")
; ("PP7", "Stated-purpose")
].
```

– subsumption of privacy preferences

```
subPurpose :: Purpose * Purpose [TRN,ASY]
PRAGMA "" "" subsumes, is less than or equally strict as ""
= [ ("General-purpose", "Treatment")
; ("General-purpose", "Insurance")
; ("General-purpose", "Research")
; ("General-purpose", "Teaching")
; ("General-purpose", "Development")
; ("General-purpose", "Marketing")
; ("Research", "Teaching")
; ("Research", "Development")
; ("Research", "Marketing")
; ("Research", "Research")
; ("Teaching", "Teaching")
; ("Development", "Development")
; ("Marketing", "Marketing")
; ("General-purpose", "General-purpose")
; ("Treatment", "Treatment")
; ("Insurance", "Insurance")
].
```

```
subRetention :: Retention * Retention [TRN,ASY]
PRAGMA "" "" subsumes, is less strict than or equally strict as ""
= [ ("Indefinitely", "Business-practice")
; ("Indefinitely", "Legal-requirement")
; ("Indefinitely", "Stated-purpose")
; ("Indefinitely", "No-retention")
; ("Business-practice", "Legal-requirement")
; ("Business-practice", "Stated-purpose")
; ("Business-practice", "No-retention")
; ("Legal-requirement", "Stated-purpose")
; ("Legal-requirement", "No-retention")
; ("Stated-purpose", "No-retention")
; ("Indefinitely", "Indefinitely")
; ("Business-practice", "Business-practice")
].
```

```

; ("Legal-requirement", "Legal-requirement")
; ("Stated-purpose", "Stated-purpose")
; ("No-retention", "No-retention")
].

```

```

subsRequestor :: Requestor * Requestor [TRN,ASY]
PRAGMA "" " subsumes, is less strict than or equally strict as "
= [ ("All", "Known-recipient")
; ("All", "Same")
; ("All", "Ours")
; ("Known-recipient", "Same")
; ("Known-recipient", "Ours")
; ("Same", "Ours")
; ("All", "All")
; ("Known-recipient", "Known-recipient")
; ("Same", "Same")
; ("Ours", "Ours")
].

```

– limited medical ontology, subsumption

```

subsData :: Data * Data [TRN,ASY]
PRAGMA "" " subsumes, is more general than or equal to "
= [ ("All-data", "Personal-data")
; ("All-data", "Medical-data")
; ("All-data", "Test-results")
; ("All-data", "Statistics")
; ("Medical-data", "Test-results")
; ("Medical-data", "Statistics")
; ("All-data", "All-data")
; ("Personal-data", "Personal-data")
; ("Medical-data", "Medical-data")
; ("Test-results", "Test-results")
; ("Statistics", "Statistics")
].

```

– basis of privacy preferences

```

belongsTo :: PrivacyPreference → Participant
PRAGMA "" " belongs to "
= [ ("PP1", "Fox")
; ("PP2", "MacBeth")
; ("PP3", "SP1-university")
; ("PP4", "SP2-hospital")
; ("PP5", "SP3-industry")
; ("PP6", "MacBeth")
; ("PP7", "SP2-hospital")
].

```

```

refersToData :: PrivacyPreference → Data

```

```

PRAGMA "" " refers to "
= [ ("PP1", "All-data")
; ("PP2", "Medical-data")
; ("PP3", "Medical-data")
; ("PP4", "Medical-data")
; ("PP5", "All-data")
; ("PP6", "Personal-data")
; ("PP7", "Personal-data")
].

```

– information requests

```
requestsInformation :: SP * PrivacyPreference
```

```
PRAGMA "" " requests data submitted to privacy preference "
```

```

= [ ("SP1-university", "PP1")
; ("SP3-industry", "PP2")
; ("SP2-hospital", "PP2") –medical data MacBeth, allowed
; ("SP2-hospital", "PP6") –personal data MacBeth, not allowed
; ("SP1-university", "PP5")
; ("SP2-hospital", "PP3")
].

```

– RULES –

```
requestsInformation -: (belongsTo~; hasPurpose; subsPurpose~; hasPurpose~) /\ (belongsTo~; refersToData; subsData~; refersToData~)
```

EXPLANATION "Information can only be requested from a party with an equally or less strict purpose policy for that type of data."

```
requestsInformation -: (belongsTo~; hasRequestor; subsRequestor~; hasRequestor~) /\ (belongsTo~; refersToData; subsData~; refersToData~)
```

EXPLANATION "Information can only be requested from a party with an equally or less strict requestor policy for that type of data."

```
requestsInformation -: (belongsTo~; hasRetention; subsRetention~; hasRetention~) /\ (belongsTo~; refersToData; subsData~; refersToData~)
```

EXPLANATION "Information can only be requested from a party with an equally or less strict retention policy for that type of data."

ENDPATTERN

ENDCONTEXT

Appendix 8 - ADL *Alternative reaction to policy*

ADL program created for this research.

CONTEXT PrivacyCompliance

```
CONCEPT "Participant" "party in federated network, person or service provider." ""
CONCEPT "Person" "person using services of federated network." ""
CONCEPT "SP" "party providing services in federated network." ""
CONCEPT "PrivacyPreference" "a policy statement how to deal with information." ""
CONCEPT "Purpose" "the purpose for which information may be used." ""
CONCEPT "Requestor" "the type of organisation which requests information." ""
CONCEPT "Retention" "the time during which information may be kept." ""
CONCEPT "Data" "the type of data that can be stored of a person." ""
CONCEPT "Permission" "permission to allow transfer of data to requestor." ""
```

PATTERN InformationRequest

– introducing sub entities

GEN Person ISA Participant

GEN SP ISA Participant

– defining privacy preferences

hasPurpose :: PrivacyPreference * Purpose [TOT]

PRAGMA "" " states that information can be used for purpose "

```
= [ ("PP1", "General-purpose")
; ("PP2", "Treatment")
; ("PP2", "Insurance")
; ("PP2", "Development")
; ("PP2", "Teaching")
; ("PP3", "Teaching")
; ("PP3", "Development")
; ("PP4", "Treatment")
; ("PP4", "Insurance")
; ("PP5", "Research")
; ("PP6", "Treatment")
; ("PP7", "Insurance")
; ("PP7", "Treatment")
].
```

hasRequestor :: PrivacyPreference * Requestor [TOT]

PRAGMA "" " states that information can be transferred to requestor "

```
= [ ("PP1", "All")
; ("PP2", "Known-recipient")
; ("PP3", "Same")
; ("PP4", "Ours")
; ("PP5", "All")
; ("PP6", "Ours")
```

```
; ("PP7", "Same")
].
```

```
hasRetention :: PrivacyPreference * Retention [TOT]
PRAGMA "" " states that information can be kept as long as needed for "
= [ ("PP1", "Indefinitely")
; ("PP2", "Legal-requirement")
; ("PP3", "Business-practice")
; ("PP4", "Stated-purpose")
; ("PP5", "Indefinitely")
; ("PP6", "No-retention")
; ("PP7", "Stated-purpose")
].
```

```
– subsumption of privacy preferences
subsPurpose :: Purpose * Purpose [TRN,ASY]
PRAGMA "" " subsumes, is less than or equally strict as "
= [ ("General-purpose", "Treatment")
; ("General-purpose", "Insurance")
; ("General-purpose", "Research")
; ("General-purpose", "Teaching")
; ("General-purpose", "Development")
; ("General-purpose", "Marketing")
; ("Research", "Teaching")
; ("Research", "Development")
; ("Research", "Marketing")
; ("Research", "Research")
; ("Teaching", "Teaching")
; ("Development", "Development")
; ("Marketing", "Marketing")
; ("General-purpose", "General-purpose")
; ("Treatment", "Treatment")
; ("Insurance", "Insurance")
].
```

```
subsRetention :: Retention * Retention [TRN,ASY]
PRAGMA "" " subsumes, is less strict than or equally strict as "
= [ ("Indefinitely", "Business-practice")
; ("Indefinitely", "Legal-requirement")
; ("Indefinitely", "Stated-purpose")
; ("Indefinitely", "No-retention")
; ("Business-practice", "Legal-requirement")
; ("Business-practice", "Stated-purpose")
; ("Business-practice", "No-retention")
; ("Legal-requirement", "Stated-purpose")
; ("Legal-requirement", "No-retention")
; ("Stated-purpose", "No-retention")
; ("Indefinitely", "Indefinitely")
].
```

```

; ("Business-practice", "Business-practice")
; ("Legal-requirement", "Legal-requirement")
; ("Stated-purpose", "Stated-purpose")
; ("No-retention", "No-retention")
].

```

```

subsRequestor :: Requestor * Requestor [TRN,ASY]
PRAGMA "" " subsumes, is less strict than or equally strict as "
= [ ("All", "Known-recipient")
; ("All", "Same")
; ("All", "Ours")
; ("Known-recipient", "Same")
; ("Known-recipient", "Ours")
; ("Same", "Ours")
; ("All", "All")
; ("Known-recipient", "Known-recipient")
; ("Same", "Same")
; ("Ours", "Ours")
].

```

– limited medical ontology, subsumption

```

subsData :: Data * Data [TRN,ASY]
PRAGMA "" " subsumes, is more general than or equal to "
= [ ("All-data", "Personal-data")
; ("All-data", "Medical-data")
; ("All-data", "Test-results")
; ("All-data", "Statistics")
; ("Medical-data", "Test-results")
; ("Medical-data", "Statistics")
; ("All-data", "All-data")
; ("Personal-data", "Personal-data")
; ("Medical-data", "Medical-data")
; ("Test-results", "Test-results")
; ("Statistics", "Statistics")
].

```

– basis of privacy preferences

```

belongsTo :: PrivacyPreference → Participant
PRAGMA "" " belongs to "
= [ ("PP1", "Fox")
; ("PP2", "MacBeth")
; ("PP3", "SP1-university")
; ("PP4", "SP2-hospital")
; ("PP5", "SP3-industry")
; ("PP6", "MacBeth")
; ("PP7", "SP2-hospital")
].

```

```

refersToData :: PrivacyPreference → Data
PRAGMA "" " refers to "
= [ ("PP1", "All-data")
; ("PP2", "Medical-data")
; ("PP3", "Medical-data")
; ("PP4", "Medical-data")
; ("PP5", "All-data")
; ("PP6", "Personal-data")
; ("PP7", "Personal-data")
].

```

```

–allowing transfer of data when special permission is given
permissionConcerns :: Permission → PrivacyPreference
PRAGMA "" " covers permission to release data overruling privacy preference "
= [ ("Perm1", "PP3")
].

```

```

permissionTo :: Permission → SP
PRAGMA "" " gives permission to release data overruling privacy preference to "
= [ ("Perm1", "SP2-hospital")
].

```

```

– information requests
requestsInformation :: SP * PrivacyPreference
PRAGMA "" " requests data submitted to privacy preference "
= [ ("SP1-university", "PP1")
; ("SP3-industry", "PP2")
; ("SP2-hospital", "PP2") –medical data MacBeth, allowed
; ("SP2-hospital", "PP6") –personal data MacBeth, not allowed
; ("SP1-university", "PP5")
; ("SP2-hospital", "PP3")
].

```

– RULES –

```

requestsInformation -: ((belongsTo~; hasPurpose; subsPurpose~; hasPurpose) /\ (belongsTo~; refersToData; subsData~; refersToData)) \/ (permissionTo~; permissionConcerns)
EXPLANATION "Information can only be requested from a party with an equally or less strict purpose policy for that type of data unless special permission was given."

```

```

requestsInformation -: ((belongsTo~; hasRequestor; subsRequestor~; hasRequestor~) /\ (belongsTo~; refersToData; subsData~; refersToData~)) \/ (permissionTo~; permissionConcerns)
EXPLANATION "Information can only be requested from a party with an equally or less strict requestor policy for that type of data unless special permission was given."

```

```

requestsInformation -: ((belongsTo~; hasRetention; subsRetention~; hasRetention~) /\ (belongsTo~; refersToData; subsData~; refersToData~)) \/ (permissionTo~; permissionConcerns)

```

cerns)

EXPLANATION "Information can only be requested from a party with an equally or less strict retention policy for that type of data unless special permission was given."

ENDPATTERN

ENDCONTEXT

Appendix 9 - ADL *Check logging privacy compliance*

ADL program created for this research.

CONTEXT PrivacyCompliance

```
CONCEPT "Participant" "party in federated network, person or service provider." ""
CONCEPT "Person" "person using services of federated network." ""
CONCEPT "SP" "party providing services in federated network." ""
CONCEPT "PrivacyPreference" "a policy statement how to deal with information." ""
CONCEPT "Purpose" "the purpose for which information may be used." ""
CONCEPT "Requestor" "the type of organisation which requests information." ""
CONCEPT "Retention" "the time during which information may be kept." ""
CONCEPT "Data" "the type of data that can be stored of a person." ""
CONCEPT "Permission" "permission to allow transfer of data to requestor." ""
CONCEPT "Logging" "entry in logging table of data request." ""
CONCEPT "Timestamp" "Time when logging was entered" ""
```

PATTERN InformationRequest

– introducing sub entities

GEN Person ISA Participant

GEN SP ISA Participant

– defining privacy preferences

hasPurpose :: PrivacyPreference * Purpose [TOT]

PRAGMA "" " states that information can be used for purpose "

```
= [ ("PP1", "General-purpose")
; ("PP2", "Treatment")
; ("PP2", "Insurance")
; ("PP2", "Development")
; ("PP2", "Teaching")
; ("PP3", "Teaching")
; ("PP3", "Development")
; ("PP4", "Treatment")
; ("PP4", "Insurance")
; ("PP5", "Research")
; ("PP6", "Treatment")
; ("PP7", "Insurance")
; ("PP7", "Treatment")
].
```

hasRequestor :: PrivacyPreference * Requestor [TOT]

PRAGMA "" " states that information can be transferred to requestor "

```
= [ ("PP1", "All")
; ("PP2", "Known-recipient")
; ("PP3", "Same")
; ("PP4", "Ours")
```

```

; ("PP5", "All")
; ("PP6", "Ours")
; ("PP7", "Same")
].

```

```

hasRetention :: PrivacyPreference * Retention [TOT]
PRAGMA "" " states that information can be kept as long as needed for "
= [ ("PP1", "Indefinitely")
; ("PP2", "Legal-requirement")
; ("PP3", "Business-practice")
; ("PP4", "Stated-purpose")
; ("PP5", "Indefinitely")
; ("PP6", "No-retention")
; ("PP7", "Stated-purpose")
].

```

```

– subsumption of privacy preferences
subsPurpose :: Purpose * Purpose [TRN,ASY]
PRAGMA "" " subsumes, is less than or equally strict as "
= [ ("General-purpose", "Treatment")
; ("General-purpose", "Insurance")
; ("General-purpose", "Research")
; ("General-purpose", "Teaching")
; ("General-purpose", "Development")
; ("General-purpose", "Marketing")
; ("Research", "Teaching")
; ("Research", "Development")
; ("Research", "Marketing")
; ("Research", "Research")
; ("Teaching", "Teaching")
; ("Development", "Development")
; ("Marketing", "Marketing")
; ("General-purpose", "General-purpose")
; ("Treatment", "Treatment")
; ("Insurance", "Insurance")
].

```

```

subsRetention :: Retention * Retention [TRN,ASY]
PRAGMA "" " subsumes, is less strict than or equally strict as "
= [ ("Indefinitely", "Business-practice")
; ("Indefinitely", "Legal-requirement")
; ("Indefinitely", "Stated-purpose")
; ("Indefinitely", "No-retention")
; ("Business-practice", "Legal-requirement")
; ("Business-practice", "Stated-purpose")
; ("Business-practice", "No-retention")
; ("Legal-requirement", "Stated-purpose")
; ("Legal-requirement", "No-retention")
].

```

```

; ("Stated-purpose", "No-retention")
; ("Indefinitely", "Indefinitely")
; ("Business-practice", "Business-practice")
; ("Legal-requirement", "Legal-requirement")
; ("Stated-purpose", "Stated-purpose")
; ("No-retention", "No-retention")
].

```

```

subsRequestor :: Requestor * Requestor [TRN,ASY]
PRAGMA "" " subsumes, is less strict than or equally strict as "
= [ ("All", "Known-recipient")
; ("All", "Same")
; ("All", "Ours")
; ("Known-recipient", "Same")
; ("Known-recipient", "Ours")
; ("Same", "Ours")
; ("All", "All")
; ("Known-recipient", "Known-recipient")
; ("Same", "Same")
; ("Ours", "Ours")
].

```

– limited medical ontology, subsumption

```

subsData :: Data * Data [TRN,ASY]
PRAGMA "" " subsumes, is more general than or equal to "
= [ ("All-data", "Personal-data")
; ("All-data", "Medical-data")
; ("All-data", "Test-results")
; ("All-data", "Statistics")
; ("Medical-data", "Test-results")
; ("Medical-data", "Statistics")
; ("All-data", "All-data")
; ("Personal-data", "Personal-data")
; ("Medical-data", "Medical-data")
; ("Test-results", "Test-results")
; ("Statistics", "Statistics")
].

```

– basis of privacy preferences

```

belongsTo :: PrivacyPreference → Participant
PRAGMA "" " belongs to "
= [ ("PP1", "Fox")
; ("PP2", "MacBeth")
; ("PP3", "SP1-university")
; ("PP4", "SP2-hospital")
; ("PP5", "SP3-industry")
; ("PP6", "MacBeth")
; ("PP7", "SP2-hospital")
].

```

].

refersToData :: PrivacyPreference → Data

PRAGMA "" " refers to "

```
= [ ("PP1", "All-data")
; ("PP2", "Medical-data")
; ("PP3", "Medical-data")
; ("PP4", "Medical-data")
; ("PP5", "All-data")
; ("PP6", "Personal-data")
; ("PP7", "Personal-data")
].
```

– allowing transfer of data when special permission is given

permissionConcerns :: Permission → PrivacyPreference

PRAGMA "" " covers permission to release data overruling privacy preference "

```
= [ ("Perm1", "PP3")
].
```

permissionTo :: Permission → SP

PRAGMA "" " gives permission to release data overruling privacy preference to "

```
= [ ("Perm1", "SP2-hospital")
].
```

– logged requests

logConcernsPP :: Logging → PrivacyPreference

PRAGMA "" " logging concerns data covered by privacy preference "

```
= [ ("Log1", "PP1")
; ("Log2", "PP2")
; ("Log3", "PP2")
; ("Log4", "PP6")
; ("Log5", "PP5")
; ("Log6", "PP3")
].
```

logConcernsRequestor :: Logging → SP

PRAGMA "" " logging concerns data requested by "

```
= [ ("Log1", "SP1-university")
; ("Log2", "SP1-university")
; ("Log3", "SP2-hospital")
; ("Log4", "SP2-hospital")
; ("Log5", "SP1-university")
; ("Log6", "SP2-hospital")
].
```

loggedAt :: Logging → Timestamp

PRAGMA "" " logging entered at "

```
= [ ("Log1", "20080115")
```

```
; ("Log2", "20071218")
; ("Log3", "20080510")
; ("Log4", "20071127")
; ("Log5", "20080528")
; ("Log6", "20080601")
].
```

– RULES –

```
logConcernsRequestor~; logConcernsPP -: ((belongsTo~; hasPurpose; subsPurpose~; hasPurpose ) /\ (belongsTo~; refersToData; subsData~; refersToData )) \\/ (permissionTo~; permissionConcerns)
```

EXPLANATION "Information can only be requested from a party with an equally or less strict purpose policy for that type of data unless special permission was given."

```
logConcernsRequestor~; logConcernsPP -: ((belongsTo~; hasRequestor; subsRequestor~; hasRequestor~) /\ (belongsTo~; refersToData; subsData~; refersToData~)) \\/ (permissionTo~; permissionConcerns)
```

EXPLANATION "Information can only be requested from a party with an equally or less strict requestor policy for that type of data unless special permission was given."

```
logConcernsRequestor~; logConcernsPP -: ((belongsTo~; hasRetention; subsRetention~; hasRetention~) /\ (belongsTo~; refersToData; subsData~; refersToData~)) \\/ (permissionTo~; permissionConcerns)
```

EXPLANATION "Information can only be requested from a party with an equally or less strict retention policy for that type of data unless special permission was given."

```
ENDPATTERN
ENDCONTEXT
```


Appendix 10 - Rules *Check logging privacy compliance*

Specifications generated with ADL program created for this research.

1. Information can only be requested from a party with an equally or less strict purpose policy for that type of data unless special permission was given.
2. Information can only be requested from a party with an equally or less strict requestor policy for that type of data unless special permission was given.
3. Information can only be requested from a party with an equally or less strict retention policy for that type of data unless special permission was given.
4. Elke privacyPreference states that information can be used for purpose ten minste een purpose.
5. Elke privacyPreference states that information can be transferred to requestor ten minste een requestor.
6. Elke privacyPreference states that information can be kept as long as needed for ten minste een retention.
7. subsPurpose is een ordeningsrelatie tussen purposes.
8. subsRetention is een ordeningsrelatie tussen retentionen.
9. subsRequestor is een ordeningsrelatie tussen requestoren.
10. subsData is een ordeningsrelatie tussen datas.
11. Elke privacyPreference belongs to precies één participant.
12. Elke privacyPreference refers to precies één data.
13. Elke permission covers permission to release data overruling privacy preference precies één privacyPreference.
14. Elke permission gives permission to release data overruling privacy preference to precies één sP.
15. Elke logging logging concerns data covered by privacy preference precies één privacyPreference.
16. Elke logging logging concerns data requestes by precies één sP.
17. Elke logging logging entered at precies één timestamp.