

## D2.1

Version: 1.0  
Date: 2008-07-31  
Author: TILBURG  
UNIVERSITY

Dissemination status: PU  
Document reference: D2.1



# State-of-the-art in the field of compliance languages

Project acronym:	COMPAS	
Project name:	Compliance-driven Models, Languages, and Architectures for Services	
Call and Contract:	FP7-ICT-2007-1	
Grant agreement no.:	215175	
Project Duration:	01.02.2008 – 28.02.2011 (36 months)	
Co-ordinator:	TUV	Vienna University of Technology (AT)
Partners:	CWI	Stichting Centrum voor Wiskunde en Informatica (NL)
	UCBL	Université Claude Bernard Lyon 1 (FR)
	USTUTT	Universitaet Stuttgart (DE)
	TILBURG UNIVERSITY	Stichting Katholieke Universiteit Brabant (NL)
	UNITN	Universita degli Studi di Trento (IT)
	TARC-PL	Apera sp. z o.o. (PL)
	THALES	Thales Services SAS (FR)
	PWC	Pricewaterhousecoopers Accountants N.V. (NL)

This project is supported by funding from the Information Society Technologies Programme under the 7th Research Framework Programme of the European Union.





Project no. 215175

**COMPAS**

**Compliance-driven Models, Languages, and Architectures for Services**

Specific Targeted Research Project

Information Society Technologies

## **D2.1 State-of-the-art in the field of compliance languages**

Due date of deliverable: 2008-07-31

Actual submission date: 2008-07-31

Start date of project: 2008-02-01      Duration: 36 months

Organisation name of lead partner for this deliverable: TILBURG UNIVERSITY  
Tilburg University, The Netherlands

Revision 1.0

<b>Project funded by the European Commission within the Seventh Framework Programme</b>		
<b>Dissemination Level</b>		
<b>PU</b>	Public	X
<b>PP</b>	Restricted to other programme participants (including the Commission Services)	
<b>RE</b>	Restricted to a group specified by the consortium (including the Commission Services)	
<b>CO</b>	Confidential, only for members of the consortium (including the Commission Services)	

## History chart

Issue	Date	Changed page(s)	Cause of change	Implemented by
0.1	2008-04-18	All sections	New document	TILBURG UNIVERSITY
0.2	2008-04-25	Content added to section 2	Revised document	TILBURG UNIVERSITY
0.4	2008-05-08	Content added to section 3	Revised document	TILBURG UNIVERSITY
0.6	2008-05-20	Content added to section 4	Revised document	TILBURG UNIVERSITY
0.7	2008-06-01	Content added to section 5	Revised document	TILBURG UNIVERSITY
0.8	2008-06-09	References added	Revised document	TILBURG UNIVERSITY
0.9	2008-06-14	Internal review	Revised document	TILBURG UNIVERSITY
0.95	2008-06-18	Revision based on internal review results	Revised document	TILBURG UNIVERSITY
0.98	2008-07-08	Revision after partner reviews	Revised document	TILBURG UNIVERSITY
0.99	2008-07-21	Shortened after partner suggestions	Revised document	TILBURG UNIVERSITY
1.0	2008-07-29	Incorporation of final comments and last touch up	Revised document	TILBURG UNIVERSITY

## Authorisation

No.	Action	Company/Name	Date
1	Prepared	TILBURG UNIVERSITY	2008-07-29
2	Approved	TUV	2008-07-31
3	Released	TUV	2008-07-31

Disclaimer: The information in this document is subject to change without notice. Company or product names mentioned in this document may be trademarks or registered trademarks of their respective companies.

### All rights reserved.

This document is proprietary of the COMPAS consortium members. The document is supplied on the express understanding that it is to be treated as confidential and may not be used or disclosed to others in whole or in part for any purpose except as expressly authorised in terms of EC contract number 215175.

## Contents

1. Introduction .....	7
1.1. Purpose and scope .....	8
1.2. Document overview .....	9
1.3. Reference documents .....	9
1.3.1. Internal documents.....	9
1.3.2. External documents .....	9
1.4. Definitions and glossary.....	16
1.5. Abbreviations and acronyms.....	16
2. Compliance Frameworks.....	19
2.1. Compliance discovery .....	19
2.2. Compliance specification .....	20
2.3. Compliance reporting.....	21
2.4. Compliance improvement .....	22
2.5. Compliance risk management.....	23
2.6. Conclusions .....	23
3. Compliance discovery .....	25
3.1. Types Of Compliance Concerns .....	26
3.2. Types Of Affected Business Processes .....	29
3.3. Conclusions .....	30
4. Compliance specification .....	31
4.1. Modelling basic compliance concerns .....	31
4.1.1. Business process modelling approaches from industry .....	31
4.1.1.1. SOA and workflow based business process modelling .....	31
4.1.1.2. Generic business process modelling approaches .....	32
4.1.2. Business process modelling approaches from academia .....	33
4.1.3. Advanced business process modelling approaches .....	34
4.1.4. Tool support.....	35
4.2. Modelling advanced compliance concerns .....	36
4.2.1. Monitoring .....	36
4.2.1.1. Definition .....	37
4.2.1.2. Detection .....	39
4.2.1.3. Notification .....	39
4.2.1.4. Tool support.....	41

4.2.2.	Payment .....	41
4.2.3.	Privacy .....	42
4.2.4.	Quality .....	44
4.2.5.	Retention.....	45
4.2.6.	Security .....	47
4.2.7.	Transaction .....	48
4.3.	Conclusions .....	50
5.	Compliance specification and improvement .....	52
5.2.	Rule based approaches .....	52
5.2.1.	Specification .....	52
5.2.2.	Analysis .....	56
5.2.3.	Implementation .....	58
5.3.	Policy based approaches .....	59
5.4.	Goal based approaches .....	62
5.5.	Conclusions .....	65
6.	Overall Conclusions .....	67

## List of figures

Figure 1: Components of a Compliance Program.....	19
Figure 2: Legislative Compliance Concerns for Business Processes.....	30
Figure 3: Interesting Features for Compliance Requirement Specification.....	66

## List of tables

Table 1: Compliance Legislation by Compliance Requirement Type .....	26
Table 2: Impact of Compliance Legislation on Business Processes .....	29

## Abstract

This deliverable provides an overview of the state-of-the-art in compliance languages, particularly focusing on languages for regulatory and legislative provisions. The deliverable first identifies different aspects of compliance, being compliance discovery, specification, reporting, improvement and risk management. Current compliance-specific solutions addressing the different aspects of compliance are then reviewed. The reviewed solutions advocate the same conceptual approach with regard to compliance specification: 1) the need for models to describe compliance enriched business processes; 2) the need for policies and rules to describe the compliance requirements to which business processes must conform; and 3) the explicit separation of models and policies while at the same time establishing how they are related. The solutions also share a weakness in the sense that they only cater for reactive compliance. That is, they only allow to determine after design whether the resulting business process model is compliant. This is troublesome in the sense that the applicability of compliance requirements is often dependent on runtime conditions.

Next, in line with the purpose and scope of the deliverable, a wide range of compliance legislations is discussed. Specifically, the legislations defined within Basel II, FINRA (NASD/SEC) regulations, HIPAA, IFRS, MIFID, Sarbanes-Oxley and Tabaksblat are analysed. This set of regulations constitutes a faithful representation of the range of compliance requirements that can be found within compliance legislations. They are also in sync with the case studies carried out within the COMPAS project. From this analysis a set of core compliance concerns is identified. One group of core concerns pertains to the basic structure of business processes, and relates to their control flow, information usage, location related details, employed resources and temporal occurrences. These are referred to as basic compliance concerns. The second group of core concerns is more advanced in nature, and builds on these basic concerns. Specifically, the monitoring, payment, privacy, quality, retention, security, and transaction concern are identified; and it is established how these build on basic concern(s). Moreover, it is investigated how these advanced concerns relate to each other (e.g. when information has to be retained in a secure manner). Finally, the open issue of how to combine these core concerns into more high level ones is mentioned; e.g. to define an auditing concern comprising (among others) logging and retention, and security.

Existing solutions for modelling these concerns are then analysed. Concretely, a variety of business process modelling languages is surveyed to examine their support for the expression of basic compliance concerns. Next, for each of the advanced compliance concerns relevant works are investigated. The results form a point of departure for the development of views for the identified compliance concerns using the Model-driven Integration Architecture for Compliance. Particularly, it is likely that each concern will lead to description within process views, where the view constructs will be defined within one or more domain specific languages (DSLs). After that, a variety of compliance requirement specification languages are examined. Features in four categories are singled out, being heterogeneity (e.g. uniform specification and symmetry of specification), expressive power (e.g. conceptually natural semantics and prioritization), manageability (e.g. documentation and life cycle management) and usability (e.g. customizable and interactive). They will be used as a point of reference during the development of the compliance request language within the COMPAS project. Furthermore, the synergy among goal, policy and rule specification is explored to create a tentative method in which compliance goals capture abstract compliance requirements for business processes by referring to compliance legislations and process characteristics. These then lead to the identification of compliance policies containing applicable compliance rules; which can then be applied to business processes. This forms an interesting avenue of departure for the construction and usage of the compliance request language.

# 1. Introduction

Today's business climate demands a high rate of compliance of business processes with which Information Technology (IT)-minded organizations are required to cope. Compliance regulations, such as HIPAA, Basel II, Sarbanes-Oxley (SOX) and others require all organizations to review their business processes and ensure that they meet the compliance standards set forth in the legislation. This includes, but is not limited to, data acquisition and archival, document management, data security, financial accounting practices, shareholder reporting functions and to know when unusual activities occur. In a broader perspective compliance can pertain to any explicitly stated rule or regulation that prescribes any aspect of an internal or cross-organizational business process; including for example public policies, customer preferences, partner agreements and jurisdictional provisions.

Currently compliance to such rules and regulations is typically achieved on a per-case basis. Often compliance solutions are hand crafted for particular compliance problems. From a management perspective they have several undesirable characteristics: 1) they are hard to maintain as they do not follow a well established architectural pattern; 2) they are hard to evolve as the solutions usually involve hard coding requirements across multiple systems with ill-defined dependencies among components; 3) they are hard to reuse since they were custom made for particular compliance problems; 4) they are hard to understand because a compliance solutions often addresses highly intertwined compliance requirements; and 5) it is difficult to ensure guaranteed compliance, i.e. formally verifiable.

In addition to this list of problems of current compliance practice, another objection is that it inhibits organizations from easily making changes to their compliance solutions. This is a major concern, as organizations are facing rapidly changing market conditions, new competitive pressures, new regulatory fiats, new competitive threats and so on. All of these changes drive the need for the IT infrastructure of an organization to respond quickly in support of new business models and requirements. Organizations must therefore be agile and dynamic, for only in this way can they gear towards the world of fast occurring, semi-automated, and complex electronic transactions. This leads to a nightmarish scenario in which organizations have to invest heavily in regularly updating customized compliance solutions.

In response to these developments many Information Technology (IT)-minded organizations have adopted the service oriented computing paradigm in order to realize corporate business services that can easily adapt to changes. This paradigm promises to deliver the ability to dynamically and rapidly grow application portfolios by assembling new services to support organizational business processes. The resulting SOA-enabled business processes (SEPs) typically stretch across a diverse range of cooperating and coordinated systems, often crossing organizational boundaries. Popular technologies to plumb the integration and interactions of these (typically) heterogeneous systems include XML and SOAP, whereas technologies such as WSDL, WS-BPEL and Web Service Choreography Description Language are utilised to describe, orchestrate and choreograph services in a standardised manner; together encompassing comprehensive descriptions of service enabled business processes.

Unfortunately, although these technologies bring closer the promise of more agile and adaptive business processes, they are lacking when it comes to offering support for compliance. Typically, compliance requirements not only pertain to the basic functional structures of processes, but more often than not to extra-functional concerns such as monitoring and notification of unusual events (real-time awareness), quality of information and services (e.g. financial data), retention and archiving of messages and events, security of access and communication, and preservation of transactional semantics. In order to bring

about verifiable compliance of business processes to such compliance concerns it must be possible to define and subsequently integrate these specifics into service enabled business process models. This necessitates the need for so-called compliance concern languages that identify and describe the constructs facilitating such definition. Particularly, different compliance legislations bring with them different concerns and thus necessitate different modelling constructs for business processes.

## 1.1. Purpose and scope

In the European Commission 7th framework COMPAS project on compliance-driven models, languages, and architectures for services, models, languages, and an architectural framework are being developed including required software components and services to ensure dynamic and on-going compliance of software services to business regulations (e.g. Basel II, IFRS and Sarbanes-Oxley) and the user service requirements. As part of this work, this deliverable presents the state-of-the-art in compliance languages, especially focusing on requirements found in compliance legislations (and similar domain oriented concerns).

Specifically, the purpose of the report is to identify measurable properties of business processes and protocols, and expressiveness, changeability and reuse in existing compliance languages. To this end the report discusses a wide range of compliance legislations of relevance for business process compliance in order to identify common underlying compliance concerns. This will result in establishing a clear view on the types of compliance requirements that will be addressed within the COMPAS project.

Next, existing solutions for modelling these concerns will be analysed and measurable properties of business processes and protocols extrapolated. Particularly, in this regard there is a strong emphasis within the report on the distillation and description of properties in current solutions targeted at automated business processes; given the overall focus of the COMPAS project on the compliance of service enabled processes.

After that a variety of compliance requirement specification languages are examined including works from the fields of policy, rule and goal description. During this examination potentially useful features of the proposed approaches for compliance are identified. In these review processes the intent is on surveying the existing solution space in the field of compliance languages and business process compliance to identify potential strengths as well as current weaknesses (rather than provide an exhaustive overview).

This deliverable is related to Deliverable 5.1 [D5.1] developed in work package 5, which describes the state-of-the-art in service composition monitoring and management. While this deliverable focuses on the monitoring in order to understand how to best formalize and specify events, Deliverable 5.1 looks at the monitoring problem in order to understand how to best observe these events, how to warehouse them, and how to present them to the human user. As such, this deliverable and Deliverable 5.1 are complimentary in nature.

Also, this deliverable examines a variety of compliance legislations and regulations that are part of the case studies defined and implemented within the COMPAS project to validate the developed methodologies, techniques. Concretely, in relation to Deliverable 6.1 [D6.1] the case studies presented by Thales and PricewaterhouseCoopers provide real life examples of requirements flowing from regulations and codes such as Basel II, COBIT, COSO, IFRS, Sarbanes-Oxley, and Tabaksblatt. The case studies developed by Telcordia Poland are complementary to this deliverable in the sense that they do not emphasize the regulatory component of compliance. At the same time though the compliance requirements put forward in these case studies are very similar in spirit to those found in compliance legislations.

## 1.2. Document overview

The deliverable is structured as follows: first in section 2 recent frameworks for business process compliance are discussed. From this discussion several aspects of compliance are identified, being discovery, specification, reporting, improvement and risk management. Conform the purpose and scope of this deliverable only the discovery, specification and improvement aspects are addressed. Concretely, section 3 deals with compliance discovery by providing an analysis of a wide variety of compliance legislations in context of business processes resulting in a list of shared concerns underlying these legislations. Next, section 4 surveys existing approaches for modelling the identified compliance concerns to cater for compliance specification. After that, section 5 completes the analysis for compliance specification as well as analyses existing relevant solutions for compliance improvement. Finally, section 6 presents conclusions including an outlook for how existing work may be utilised within the COMPAS project with regard to developing methodologies, methods, techniques and tools for supporting the specification of compliance requirements.

## 1.3. Reference documents

### 1.3.1. Internal documents

- [DOW] Description of Work
- [D5.1] State-of-art in the field of Adaptive Service Composition Monitoring and Management
- [D6.1] Use Case, Metrics and Case Study

### 1.3.2. External documents

- [ABW04] G. Antoniou, A. Bikakis, and G. Wagner, "A system for non-monotonic rules on the web," in Proceedings of the RuleML 2004 conference, November 2004.
- [ADH+03] W. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. J. M. M. Weijters, "Workflow mining: a survey of issues and approaches," Data and Knowledge Engineering, vol. 47, no. 2, pp. 237–267, 2003.
- [AMP94] A. Anton, W. McCracken, and C. Potts, "Goal decomposition and scenario analysis in business process re-engineering," in Proceedings of the International Conference on Advanced Information Systems Engineering, 1994, DOI: 10.1007/3-540-58113-8\_164.
- [AND04] A. Anderson, "An introduction to the web services policy language (WSPL)," in Proceedings of the 5th IEEE International Workshop on Policies for Distributed Systems and Networks, June 2004, DOI: 10.1109/POLICY.2004.1309166.
- [APT03] K. Apt, Principles of Constraint Programming. Cambridge University Press, 2003, DOI: 10.1017/S1471068404232185.
- [ARB04] F. Arbab, "Reo: a Channel-based Coordination Model for Component Composition," Mathematical Structures in Computer Science, vol. 14, no. 3, pp. 329–366, 2004, DOI: 10.1017/S0960129504004153.

- [BA05] T. Breaux and A. Anton, “Analyzing goal semantics for rights, permissions, and obligations”, in Proceedings of the 13<sup>th</sup> IEEE International Conference on Requirements Engineering, 2005, DOI: 10.1109/RE.2005.12.
- [BAS06] T. Breaux, A. Anton, and E. Spafford, “A distributed requirements management framework for legal compliance and accountability”, North Carolina State University Computer Science, Tech. Rep. 14, 2006.
- [BCV06] S. Bleistein, K. Cox, and J. Verner, “Validating strategic alignment of organisational it requirements using goal modelling and problem diagrams,” Journal of System Software, vol. 79, no. 3, pp. 362–378, 2006, DOI: 10.1016/j.jss.2005.04.033.
- [BJ07] M. Braem and N. Joncheer, “Requirements for applying aspect-oriented techniques in web service composition languages,” in Software Composition, pp. 152–159, 2007.
- [BJP+07] M. Braem, N. Joncheere, W. van der Perren, R. V. D. Straeten, and V. Jonckers, “Concern-specific languages in a visual web service creation environment,” Electronic Notes in Theoretical Computer Science, vol. 163, pp. 3–17, 2007 DOI: 10.1016/j.entcs.2006.10.012.
- [BLR+05] A. Bandara, E. Lupu, A. Russo, N. Dulay, M. Sloman, P. Flegkas, M. Charalambides, and G. Pavlou, “Policy refinement for DiffServ quality of service management,” in Proceedings of the IEEE International Symposium on Integrated Network Management, 2005, pp. 469–482, DOI: 10.1109/INM.2005.1440817.
- [BO05] P. Bonatti and D. Olmedilla, “Reverse policy specification,” Universita di Napoli Federico II, Tech. Rep., February 2005.
- [BOL06] H. Boley, “The RuleM family of web rule languages,” in Proceedings of the International Workshop Principles and Practice of Semantic Web Reasoning, June 2006.
- [BPS01] J. Bergstra, A. Ponse, and S. Smolka, Handbook of Process Algebra. Elsevier, 2001.
- [BPW02] J. Bailey, A. Poulouvasilis, and P. Wood, “An Event-Condition-Action Language for XML,” in Proceedings of the 11<sup>th</sup> International Conference on the World Wide Web, May 2002, DOI: 10.1145/511446.511509.
- [BRJ98] G. Booch, J. Rumbaugh, and I. Jacobson, The Unified Modeling Language User Guide. Addison-Wesley, 1998.
- [BVJ+06] M. Braem, K. Verlaenen, N. Joncheere, W. Vanderperren, R. V. D. Straeten, E. Truyen, W. Joosen, and V. Jonckers, “Isolating process-level concerns using Patus,” in Proceedings of the International Conference on Business Process Management, 2006.
- [CFH06] B. Carminati, E. Ferrari, and P. Hung, Privacy Issues in the Web Services Architecture (WSA). Idea Group Inc, 2006.
- [CH01] Q. Chen and M. Hsu, “Inter-enterprise collaborative business process management,” in Proceedings of The International Conference on Data Engineering, 2001, DOI: 10.1109/ICDE.2001.914836.

- [CM07] A. Charfi and M. Mezini, "AO4BPEL: An aspect-oriented extension to BPEL," *World Wide Web*, vol. 10, pp. 309–344, 2007, DOI: 10.1007/s11280-006-0016-3.
- [CP05] M. Comuzzi and B. Pernici, "An architecture for flexible web service QoS negotiation," in *Proceedings of the Ninth IEEE International EDOC Enterprise Computing Conference*, 2005, DOI: 10.1109/EDOC.2005.4.
- [CSM+04] J. Cardoso, A. Sheth, J. Miller, J. Arnold, and K. Kochut, "Quality of service for workflows and web service processes," *Journal of Web Semantics*, vol. 1, no. 3, pp. 166–206, 2004.
- [CSM07] A. Charfi, B. Schmeling, and M. Mezini, "Transactional BPEL processes with AO4BPEL aspects," in *Proceedings of the Fifth European Conference on Web Services*, 2007, DOI: 10.1109/ECOWS.2007.29.
- [DDK+04] A. Dan, D. Davis, R. Kearney, A. Keller, R. King, D. Kuebler, H. Ludwig, M. Polan, M. Spreitzer, and A. Youssef, "Web services on demand: WSLA-driven automated management," *IBM Systems Journal*, vol. 43, no. 1, pp. 136–159, 2004.
- [DDL+00] N. Damianou, N. Dulay, E. Lupu, and M. Sloman, "Ponder: A language for specifying security and management policies for distributed systems," *Imperial College Of Science, Technology and Medicine, Tech. Rep.*, October 2000.
- [DDM+98] E. Darimont, P. Delor, P. Massonet, and A. van Lamsweerde, "An environment for goal-driven requirements engineering," in *Proceedings of the 20th International Conference on Software Engineering*, 1998.
- [DLF93] A. Dardenne, A. van Lamsweerde, and S. Fickas, "Goal-directed requirements acquisition," *Science of Computer Programming*, vol. 20, pp. 3–50, 1993, DOI: 10.1016/0167-6423(93)90021-G.
- [FK98] S. Froland and J. Koistinen, "Quality-of-service specification in distributed object systems," *Distributed System Engineering Journal*, vol. 5, no. 4, pp. 179–202, 1998.
- [GAP07] S. Ghanavati, D. Amyot, and L. Peyton, "A requirements management framework for privacy compliance," in *Proceedings of the Workshop on Requirements Engineering*, 2007.
- [GK07] A. Ghose and G. Koliadis, "Auditing business process compliance," in *Proceedings of the International Conference on Service-Oriented Computing*, 2007, DOI: 10.1007/978-3-540-74974-5\_14.
- [GM06] G. Governatori and Z. Milosevic, "A formal analysis of a business contract language," *International Journal of Cooperative Information Systems*, vol. 15, no. 4, pp. 659–685, 2006.
- [GMS+07] G. Governatori, Z. Milosevic, S. Sadiq, and M. Orłowska, "On compliance of business processes with business contracts", *Tech. Rep. TR-12216*, School of Information Technology and Electrical Engineering, University of Queensland, 2007.
- [GRS91] A. van Gelder, K. Ross, and J. Schlipf, "The well-founded semantics for general logic programs," *Journal of the ACM*, vol. 38, no. 3, pp. 620–650, 1991, DOI: 10.1145/116825.116838.

- [GV06] S. Goedertier and J. Vanthienen, "Designing compliant business processes with obligations and permissions," in Proceedings of the BPM 2006 Workshop, 2006, DOI: 10.1007/11837862\_2.
- [HB03] R. Hamadi and B. Benatallah, "A Petri net-based model for web service composition," in Proceedings of the 14<sup>th</sup> Australasian Database Conference, February 2003.
- [HOL03] G. Holzmann, "The SPIN Model Checker: Primer and Reference Manual," Addison-Wesley, 2003.
- [HON05] M. d'Hondt, "Hybrid Aspects for Integrating Rule-Based Knowledge and Object-Oriented Functionality," Ph.D. Dissertation, System and Software Engineering Lab, 2005, DOI: 10.1145/976270.976287.
- [ITGOV07] IT Governance Institute, "Framework for control objectives: Management guidelines and maturity models (COBIT 4.1)", 2007.
- [ITU-T03] ITU-T, "User requirements notation (URN) language requirements and framework," ITU-T Recommendation Z.150, February 2003.
- [JFN+96] N. Jennings, P. Faratin, T. Norman, P. O'Brien and B. Odgers, "Autonomous Agents for Business Process Management," International Journal of Applied Artificial Intelligence, vol. 14, no. 2, pp. 145–189, 1996.
- [JG06] D. Jobst and T. Greiner, "Modern process management with SOA, BAM and CEP - from static process models to executable workflows and monitoring on business level," in Proceedings of the 1st Event Processing Symposium, 2006.
- [JG07] C. Johnson and T. Grandison, "Compliance with data protection laws using Hippocratic database active enforcement and auditing," IBM Systems Journal, vol. 46, no. 2, pp. 255–264, 2007.
- [JSS97] S. Jajodia, P. Samarati, and V. Subrahmanian, "A logical language for expressing authorisations," in Proceedings of the IEEE Symposium on Security and Privacy, 1997.
- [KF07] L. Kagal and T. Finin, "Modeling conversation policies using permissions and obligations," Autonomous Agents and Multi-Agent Systems, vol. 14, no. 2, pp. 187–206, 2007, DOI: 10.1007/s10458-006-0013-z.
- [KFP+04] L. Kagal, T. Finin, M. Paolucci, N. Srinivasan, K. Sycara, and G. Denker, "Authorization and privacy for semantic web services," IEEE Intelligent Systems, vol. 19, no. 4, pp. 50–56, 2004, DOI: 10.1109/MIS.2004.23.
- [KG08] G. Koliadis and A. K. Ghose. "Service Compliance: Towards Electronic Compliance Programs", University of Wollongong, Tech. Rep. TR2008-01, 2008.
- [KHE+00] A. Kolber, D. Hay, C. Estep, D. Struck, G. Lam, J. Healy, J. Hall, J. Zachman, K. Healy, M. Eulenberg, N. Fishman, R. Ross, T. Moriarty, and W. Selkow, "Organisational business plans - the standard model for business rule motivation," Business Rule Group, November 2000.
- [KM07] A. Kamada and M. Mendes, "Business rules in a service development and execution environment," in Proceedings of the International Symposium on Communications and Information Technologies, 2007, DOI: 10.1109/ISCIT.2007.4392229.

- [KMK+07] A. Kokune, M. Mizuno, K. Kadoya, and S. Yamamoto, "FBCM - strategy modeling method for the validation of software requirements," *Journal of Software Engineering*, vol. 80, no. 3, pp. 314–327, 2007, DOI: 10.1016/j.jss.2006.04.035.
- [KS02] G. Karjoth and M. Schunter, "A privacy policy model for enterprises," in *Proceedings of the 15th IEEE Computer Security Foundations Workshop*, June 2002, DOI: 10.1109/CSFW.2002.1021821.
- [LAM01] A. van Lamsweerde, "Goal-oriented requirements engineering: A guided tour," in *Proceedings of the 5th IEEE International Symposium on Requirements Engineering*, 2001, DOI: 10.1109/ISRE.2001.948567.
- [LMX07] Y. Liu, S. Muller, and K. Xu, "A static compliance-checking framework for business process models," *IBM Systems Journal*, vol. 46, no. 2, pp. 335–362, 2007.
- [LPH04] K. Leune, M. Papazoglou, and W. van den Heuvel, "Specification and querying security constraints in the EFSOC framework," in *Proceedings of the 2d International Conference on Service Oriented Computing*, December 2004, DOI: 10.1145/1035167.1035186.
- [LTW02] N. Leemans, J. Treur, and M. Willems, "A Semantical Perspective on Verification of Knowledge," *Journal of Data and Knowledge Engineering*, vol. 40, no. 1, pp. 33–70, 2002, DOI: 10.1016/S0169-023X(01)00045-3.
- [LUC02] D. Luckham, *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems (Hardcover)*. Addison-Wesley Professional, 2002.
- [MCN92] J. Mylopoulos, L. Chung, and B. Nixon, "Representing and using non-functional requirements - a process-oriented approach," *IEEE Transactions on Software Engineering*, vol. 18, no. 6, pp. 483–497, 1992, DOI: 10.1109/32.142871.
- [MED93] B. Medvinsky, "Netcash: A design for practical electronic currency on the Internet." in *Proceedings of the First ACM Conference on Computer and Communications Security*, 1993, DOI: 10.1145/168588.168601.
- [MGM03] H. Mouratidis, P. Giorgini, and G. Manson, "An ontology for modelling security: The Tropos approach," in *Proceedings of the 7th International Conference on Knowledge-Based Intelligent Information & Engineering Systems*, Oxford, United Kingdom, September 2003.
- [MIL93] R. Milner, *Operational and Algebraic Semantics of Concurrent Processes*. Elsevier, 1993.
- [MK06] M. Papazoglou and B. Kratz, "A business-aware web services transactions model," in *Proceedings of the 4th International Conference on Service-Oriented Computing*, 2006.
- [MN04] J. Mendling and M. Nuetgens, "Exchanging EPC business process models with EPML," in *Proceedings of the 1st GI Workshop XML Interchange Formats for Business Process Management*, March 2004.

- [MR05] M. zur Muehlen and M. Rosemann, "Integrating risks in business process models," in Proceedings of the 16th Australasian Conference on Information Systems, Sydney, November-December 2005.
- [MW97] J. Mackie-Mason and K. White, "Evaluating and Selecting Digital Payment Mechanisms," In Interconnection and the Internet, pp.113–134, 1997.
- [NM95] B. Neuman and G. Medvinsky, "Requirements for network payment: the NetCheque perspective," in Proceedings of the 40th IEEE Computer Society International Conference, March 1995.
- [NRD06] C. Nagl, F. Rosenberg, and S. Dustdar, "Vidre - a distributed service-oriented business rule engine based on RuleML," in Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference, 2006, DOI: 10.1109/EDOC.2006.67.
- [NS08] K. Namiri and N. Stojanovic, "Towards a formal framework for business process compliance," in Proceedings of the Multikonferenz Wirtschaftsinformatik, February 2008.
- [ODH+07] C. Ouyang, M. Dumas, A. ter Hofstede, and W. van der Aalst, "Pattern-based Translation of BPMN Process Models to BPEL Web Services," International Journal of Web Services Research, vol. 5, no. 1, pp. 42–62, 2007.
- [ORR07] B. Orriens, On the Adaptive Development and Management of Business Collaborations, Center Dissertation Series, no.194, CentER, September 2007.
- [OY08a] B. Orriens and J. Yang, "On the specification of payment requirements for collaborative services," in Proceedings of the IEEE International Conference on Services Computing, 2008.
- [OY08b] B. Orriens and J. Yang, "On the specification of quality requirements for collaborative services," in Proceedings of the 12th IEEE International EDOC Conference, 2008 (to be published).
- [PAP+02] M. Papazoglou, M. Aiello, M. Pistore, and J. Yang, "XSRL: An XML web-services request language," Informatica e Telecomunicazioni, University of Trento, Tech. Rep., 2002.
- [PAP03] M. Papazoglou, "Web services and business transactions," World Wide Web, vol. 6, no. 1, pp. 49–91, 2003, DOI: 10.1023/A:1022308532661.
- [PAS+98] J. Peiro, N. Asokan, M. Steiner, and M. Waidner, "Designing a generic payment service," IBM Systems Journal, vol. 37, no. 1, pp. 72–88, 1998.
- [PAS05] A. Paschke, "RBSLA - a declarative rule-based service level agreement language based on RuleML," in Proceedings of the International Conference on Intelligent Agents, 2005.
- [PGS+06] V. Padmanabhan, G. Governatori, S. Sadiq, R. Colomb, and A. Rotolo., "Process modeling: the deontic way," in Proceedings of the Australia-Pacific Conference on Conceptual Modeling, 2006.
- [PSB92] A. Preece, R. Shinghal, and A. Batarekh, "Principles and Practice in Verifying Rule Based Systems," The Knowledge Engineering Review, vol. 7, no. 2, pp. 115–141, 1992.

- [RCP+07] J. Ramanathan, R. J. Cohen, E. Plassmann, and K. Ramamoorthy, "Role of an auditing and reporting service in compliance management," *IBM Systems Journal*, vol. 46, no. 2, pp 305–318, 2007.
- [RD05] F. Rosenberg and S. Dustdar, "Business Rules Integration in BPEL A Service-Oriented Approach," in *Proceedings of the 7th International IEEE Conference on E-Commerce Technology*, 2005, DOI: 10.1109/ICECT.2005.25.
- [RGY05] B. van der Raadt, J. Gordijn, and E. Yu, "Exploring web services from a business value perspective," in *Proceedings of the 13th IEEE International Conference on Requirements Engineering*, 2005, DOI: 10.1109/RE.2005.28.
- [RKM06] J. Rao, P. Kungas, and M. Matskin, "Composition of Semantic Web Services using Linear Logic Theorem Proving," *International Journal of Information systems*, vol. 31, no. 4–5, pp. 340–360, 2006, DOI: 10.1016/j.is.2005.02.005.
- [ROB+02] A. Rezgui, M. Ouzzani, A. Bouguettaya, and B. Medjahed, "Preserving privacy in web services," in *Proceedings of the 4th International Workshop on Web Information and Data Management*, November 2002, DOI: 10.1145/584931.584944.
- [ROS03] R. Ross, *Principles of the Business Rule Approach*. Addison-Wesley, 2003.
- [SCM+07] A. Svirskas, C. Courbis, R. Molva, and J. Bedzinskas, "Compliance proofs for collaborative interactions using aspect-oriented approach," in *Proceedings of the IEEE Congress on Services*, 2007, DOI: 10.1109/SERVICES.2007.23.
- [SEH02] J. O'Sullivan, D. Edmond, and A. ter Hofstede, "What's in a service? towards accurate description of non-functional service properties," *Distributed and Parallel Databases*, vol. 12, no. 2–3, pp. 117–133, 2002, DOI: 10.1023/A:1016547000822.
- [SGN07] S. Sadiq, G. Governatori, and K. Naimiri, "Modeling control objectives for business process compliance," in *Proceedings of the 5th International Conference on Business Process Management*, September 2007, DOI: 10.1007/978-3-540-75183-0\_12.
- [SPR95] R. Sprague, "Electronic document management: challenges and opportunities for information systems managers," *MIS Quarterly*, vol. 19, no. 1, pp. 29–49, 1995, DOI: 10.2307/249710.
- [TDT05] A. Tumer, A. Dogac, and H. Toroslu, "A semantic based user privacy protection," In *Intelligent Techniques for Web Personalization*, pp. 289–305, November 2005.
- [TIR+03] F. Tartanoglu, V. Issarny, A. Romanovsky, and N. Levy, "Coordinated Forward Error Recovery for Composite Web Services," in *Proceedings of the 22nd IEEE Symposium on Reliable Distributed Systems*, October 2003, DOI: 10.1109/RELDIS.2003.1238066.
- [TKM04] S. Tai, R. Khalaf, and T. Mikalsen, "Composition of coordinated web services," in *Proceedings of the 5<sup>th</sup> ACM/IFIP/USENIX International Conference on Middleware*, 2004.
- [TPP02] V. Tasic, B. Pagurek, and K. Patel, "WSOL: A language for the formal specification of various constraints and classes of service for web services," *Tech. Rep.*, Network Management and Artificial Intelligence Laboratory,

Carleton University, May 2002.

- [YKC+06] S. Yamamoto, H. Kaiya, K. Cox, and S. Bleistein, “Goal oriented requirements engineering: Trends and issues,” *IEICE Transactions on Information and Systems*, vol. E89-D, pp. 2701–2711, 2006, DOI: 10.1093/ietisy/e89-d.11.2701.
- [YKS06] G. Yee, L. Korba, and R. Song, “Extracting privacy requirements from legislation,” *Privacy Protection for E-Services*, 2006.
- [YL04] T. Yu and K. Lin, “The design of QoS-capable web services.” in *Proceedings of the 1st IEEE International Conference on e-Technology, e-Commerce, and e-Sciences*, 2004, DOI: 10.1109/EEE.2004.1287283.
- [YU97] E. Yu, “Towards modeling and reasoning support for early-phase requirements engineering,” in *Proceedings of the 3<sup>rd</sup> International Symposium on Requirements Engineering*, 1997, DOI: 10.1109/ISRE.1997.566873.
- [YUE87] K. Yue, “What does it mean to say that a specification is complete?” in *Proceedings of the Fourth International Workshop on Software Specification and Design*, 1987.
- [ZBL+03] L. Zeng, B. Benatallah, H. Lei, A. Ngu, D. Flaxer, and H. Chang, “Flexible Composition of Enterprise Web Services,” *Electronic Markets - The International Journal of Electronic Commerce and Business Media*, vol. 13, no. 2, pp. 141–152, 2003, DOI: 10.1080/1019678032000067190.
- [ZBN+04] L. Zeng, B. Benatallah, A. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, “QoS-aware middleware for web services composition,” *IEEE Transactions on Software Engineering*, vol. 30, no. 5, pp. 311–327, 2004, DOI: 10.1109/TSE.2004.11.

## 1.4. Definitions and glossary

## 1.5. Abbreviations and acronyms

Abbreviation	Full name
APPEL	A P3P Preference Exchange Language
BPMI	Business Process Modelling Initiative
BPMN	Business Process Modelling Notation
BPSS	ebXML Business Process Specification Schema
CBE	Common Base Event data model
CEP	Complex Event Processing
COBIT	Control Objectives for Information and related Technology

COMPAS	Compliance-driven Models, Languages, And Architectures for Services
COSO	Committee of Sponsoring Organizations
CPA	ebXML Collaboration Protocol Agreement
CPP	ebXML Collaboration Protocol Profile
CSP	Communicating Sequential Processes
CCS	Calculus of Communicating Systems
DAML-S	DARPA Agent Markup Language-S
EP-ML	Event Process Markup Language
EPAL	Enterprise Privacy Authorization Language
EPC	Event Process Chain
ESP	Event Stream Processing
FINRA	Financial Industry Regulatory Authority
HIPAA	Health Insurance Portability and Accountability Act
ICT	Information Communication Technology
IFRS	International Financial Reporting Standards
ISACA	Information Systems Audit and Control Association
ISO	International Standards Organization
KAOS	Knowledge Acquisition in autOmated Specification
MIFID	Markets in Financial Instruments Directive
Moreq2	Model Requirements for the management of electronic records
NFR	Non-Functional Requirements framework
OASIS	Organization for the Advancement of Structured Information Standards
OCL	Object Constraint Language
OMG	Object Management Group
P3P	Platform for Privacy Preferences
PCAOB	Public Company Accounting Oversight Board
QML	Quality of Service Modelling Language

RBSLA	Rule Based Service Level Agreement language
RuleML	Rule Markup Language
SBVR	Semantics of Business Vocabulary and Rules
SCA	Service Component Architecture
SCAP	Service Component Architecture Policy
SOA	Service Oriented Architecture
SOX	Sarbanes-Oxley Act
SWRL	Semantic Web Rule Language
UML	Unified Modelling Language
URN	User Requirements Notation
WRL	Web Rule Language
WS-AT	Web Services Atomic Transaction
WS-BA	Web Services Business Activity
WS-BPEL	Web Service Business Process Execution Language
WSCDL	Web service Choreography Description Language
WSCL	Web Service Choreography Language
WSDM	Web Services Distributed Management
WSLA	Web Service Level Agreement
WSML	Web Service Modeling Language
WSMO	Web Service Modeling Ontology
WSOL	Web Service Offerings Language
WSPL	Web Services Policy Language
WSQM	Web Services Quality Model
XACML	eXtensible Access Control Markup Language
XML	eXtensible Markup Language
XPDL	eXtensible Process Description Language
XSL	eXtensible Stylesheet Language

## 2. Compliance Frameworks

In the last years there has been an increase in attention paid to the role of compliance within business processes, as testified to by a growing body of work on this topic. This section presents a discussion of these works to examine different types of approach to this matter. To give structure to this discussion, the section is divided along different aspect of compliance addressed within existing works; being compliance discovery, specification, reporting, improvement and risk management. These are identified within [KG08] as making up the core components of so-called compliance programs. Such program can be thought of as a set of management processes that allow an organization to demonstrate its commitment to compliance. Figure 1 below shows a graphical overview. In the following each aspect is discussed in more detail and current work in this area is explored.

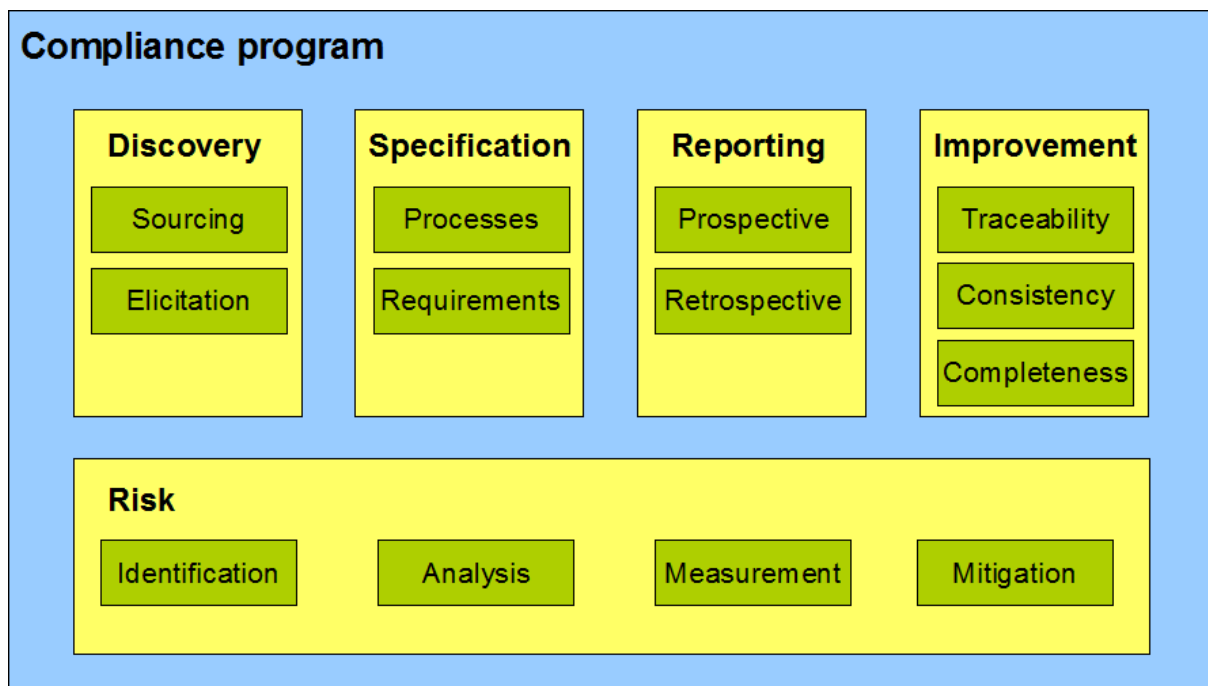


Figure 1: Components of a Compliance Program

### 2.1. Compliance discovery

Compliance discovery is concerned with finding out which compliance legislations are of importance for organizations and what the consequences are for the specification of their automated business processes (explored in Chapter 3). Several works have been produced in the area of process control objectives, attempting to charter the implications of compliance for IT. The most notably ones are COSO ([http://www.coso.org/Publications/ERM/COSO\\_ERM\\_ExecutiveSummary.pdf](http://www.coso.org/Publications/ERM/COSO_ERM_ExecutiveSummary.pdf)) and COBIT [ITGOV07]. COSO is a standard produced by the Committee of Sponsoring Organizations of the Treadway Commission. It has been promoted by the Sarbanes-Oxley oversight committee as the preferred ICT control framework. Concretely, the framework defines internal control as a process, overseen by an entity's board of directors, management and other personnel, designed to provide reasonable assurance regarding the achievement of objectives. To this end it identifies control environment, risk assessment, control activities, information and

communication, and monitoring as key components. Control activities are of relevance here as they constitute the policies and procedures that help ensure that all necessary actions are taken to address risks. Although COSO does not define how to model such activities, it does identify several of them including matters such as authorizations, data verifications, reviews of operating performance, security of assets and segregation of duties.

Related, COBIT (short for Control Objectives for Information and related Technology) provides an IT governance model to help organizations understand their IT systems. Developed by the IT Governance Institute it is aimed specifically at IT and business managers to decide the level of security and control necessary to protect their companies assets (particularly its information). The current version, COBIT 4.1, identifies thirty four high level processes that cover over two hundred control objectives categorized in four domains: planning and organization, acquisition and implementation, delivery and support, and monitoring. These objectives affect both automated and non-automated processes, and as such provide useful tools for managers, users and auditors. COBIT defines templates for how to go from IT goals by business (high level), to process controls defining what is expected to meet the high level goals, to activity goals describing how process control are implemented by application controls applicable to business applications. These fall for example in the category of management of quality of information, availability and backup of data and secure access to information. As such, COBIT is useful as it identifies a large number of control objectives for business processes, which are subsequently refined into concrete application controls. However, like COSO it does not give indication on how to model this.

Similar works have also been under development by standardizations institutes such as the OMG (<http://www.grcroundtable.org/>) and industrial companies like SAP (<http://www.sap.com/solutions/grc/index.epx>). These are all focused on defining compliance registries that contain interpretations of compliance legislations. These interpretations constitute the manner in which individual organization can apply compliance procedures (e.g. in the form of compliance policies and rules) in order to meet the obligations set forth in the legislations. These interpretations can then be used (and typically further refined) by organizations to help discover what compliance requirements their business processes have to meet. [SGN07] also advocates usage of what is called a controls directory, which holds the interpretation of compliance regulations in the specific context of an organization (given the ambiguity of such regulations). Additionally, [GK07] suggests the usage of so-called compliance patterns capturing often occurring compliance violations (e.g. separation of duty). These can aid organizations to quickly review frequent compliance violations; and in turn specify compliance requirements through compliance specification to detect and/or avoid these violations. Techniques such as these have the potential to help better facilitate the process of compliance requirement elicitation; although to be large extent this will remain the domain of compliance experts given the specific peculiarities found within individual organizations.

## 2.2. Compliance specification

Compliance specification is concerned with the modelling of compliance requirements, both within business processes as well as in the form of externalized requirements. Example work in the first area comes from [SGN07], which presents a framework for the modelling of control objectives within business process structures. Specifically, [SGN07] proposes a modal logic based approach using Formal Contract Language [GM06], which separates the prescriptive modelling of processes and the descriptive nature of compliance controls. In the approach business process models can be enriched with controls through so-called control

tags in order to realize what [SGN07] refers to as 'compliance by design'; where tags are identified affecting control flow, data, resources and time.

Another work that argues for separation of process modelling and compliance constructs modelling is [SCM+07], which proposes an aspect-oriented based approach for linking compliance to business protocols. Their idea for compliance specification is that while the protocol model is prescriptive in nature, so-called aspects are attached to them expressing different kinds of compliance requirement in a declarative manner. Inspired by aspect-oriented programming and business process modelling (discussed in more detail in section 4), aspects express requirements that are pervasive to the basic structure of a business process model (e.g. regarding security). These aspects are then attached to basic process model constructs (like activities). When the business process is to be executed, the basic protocol is expanded to include activities that enforce these aspects. Example work in this context can be found for example in [CM07].

This style of compliance modelling is also found in [LMX07], which proposes a model checking based approach for compliance specification. The paper sketches a method that allows separate modelling of business process models and compliance concerns. Business process models are expressed in the Web Services Business Process Execution Language (WS-BPEL, <http://www.oasis-open.org/committees/wsbpel/>), which are then transformed into pi-calculus and finite state machines for formalization purposes. At the same time [LMX07] is of interest as in it compliance rules are separately expressed in a graphical Business Property Specification Language. These are next translated into linear temporal logic. Then, process models are verified against the resulting statements by means of model-checking technology. [GK07] proposes a similar approach using semantically annotated process models (based on Business Process Modelling Notation (BPMN, <http://www.bpmn.org/Documents/OMG%20Final%20Adopted%20BPMN%201-0%20Spec%2006-02-01.pdf>) and formal representations of compliance requirements (defined in Computational Tree Logic) to perform compliance checking; e.g. to assess to what extent business processes deviate from applicable compliance requirements.

### 2.3. Compliance reporting

Compliance reporting is concerned with reporting on the state of compliance of business processes. [SGN07] distinguishes between two forms of compliance reporting: *retrospective reporting* and *automated detection*. Retrospective reporting constitutes after-the-fact analysis of whether or not business processes ran in compliance with regulations. As such, violations are only detected after they have occurred. Automated detection prior to compliance violations is more advanced in nature providing some level of automation to prevent actual violations from happening. [SGN07] observes further that a majority of existing compliance solutions automate some part of compliance detection by generating audit reports against specific, pre-defined checks against data pulled from enterprise applications (e.g. Enterprise Resource Planning systems). The problem is that such checks still take place after violation has occurred. [SGN07] also observes how the usage of logic based rules allows for analysing why a particular decision in a business process was made, which is of relevance for retrospective auditing reporting purposes.

At the same time the importance of retrospective compliance reporting should not be underestimated. For example, [RCP+07] discusses the role of auditing and reporting services, in which it identifies several usages of audit logs: 1) record keeping for regulatory compliance; 2) checking run-time compliance with control objectives; 3) sourcing evidence during forensic investigations; and 4) improving compliance by improving systems and

procedures. The paper also sketches an architecture for a centralized audit service that supports the creation, submission, storage, and reporting of audit data. More generic work comes from the field of process mining (see e.g. [ADH+03]). Process mining is useful as it can be applied on process event logs or real time data to monitor the behaviour of organizational business processes (and the role of humans and computerized systems therein). Possible deviations with process documentation and compliance obligations can then be detected and resolved. In the case of real time monitoring this can be done at runtime by notifying for example a human manager. Alternatively, after a business process has completed its execution its design may be modified in order to avoid future compliance violations.

## 2.4. Compliance improvement

Whereas compliance reporting is focused on establishing insight in the state of business process compliance, compliance improvement is concerned with facilitating the constant refinement of business process designs with regard to their compliance to legislations (e.g. by observing the compliance of their runtime behaviour and making adjustments accordingly). Example work is [SGN07], which notes how with its approach degrees of compliance for particular business processes can be assessed. This is of interest for compliance improvement as it allows one to assess and remedy the differences between compliance requirements and designed/actual processes. Like [SGN07], [GK07] also submits a formal mechanism for how to examine deviation of business process models to regulations in order to facilitate compliance improvement. Both works help contribute to the analysis of the *consistency* among designed processes and imposed compliance requirements. Another aspect of consistency concerns the consistency among compliance requirements. [NS08] notes for example that conflicts between requirements can occur frequently, especially in situations in which organizational business processes must be compliant to multiple compliance legislations, regulations, guidelines, standards, and so on. Although this issue has been mentioned in compliance related works, it has not been really addressed yet. [GV06] makes an attempt at ensuring *completeness* of compliance requirements, that is, to assure that all such requirements are taken into account within business process designs. To this end it describes a mechanism for generating control-flow based process models from a specification of obligations and permissions. More informally, [GMS+07] provides guidelines on how to translate contract specifications into compliant business processes designs.

A third area of compliance improvement concerns enhancement of *traceability* from compliance requirements to the design of business processes. [GAP07] suggests capturing business processes and Canadian privacy related legislative requirements separately in two models, which are then linked together (where both are specified using the User Requirements Notation (URN) [ITU-T03]). The paper then proposes to add documentation notes to the goal model capturing the privacy requirements; thus associating goals with specific legislation items. This helps to increase the traceability from compliance requirements to designed processes, which contributes positively to compliance improvement. Differently, [BAS06] observes that often compliance objectives are delegated within the organizational hierarchy, where they are refined in a top down manner. To describe this [BAS06] develops a management model consisting of actors, systems and obligations. Each actor has certain obligations (originating in this context from compliance regulations), which it is solely responsible for. To realize these obligations an actor either delegates them to other actors and/or ensures that particular systems meet those obligations. Based on this model a notion of accountability is then defined allowing one to trace which actor is responsible for which obligation and which is to be met by what system(s). In this regard it is observed that in

delegation compliance requirements are often refined. This is accommodated for through goal refinement (similar to for example what is proposed for security in [MGM03] in relation to the Tropos methodology). Delegation enhances traceability by allowing identification of responsible entities in event of compliance violations (as such aiding compliance improvement). It also improves accountability, which is useful in the context of compliance risk management.

## 2.5. Compliance risk management

Compliance risk management emphasizes the risks involved with compliance, particularly in the case when compliance demands cannot be met. Such compliance risks are typically created and defined in compliance legislations. COSO identifies external, internal and change related factors in this regard. External factors are those beyond the control of an organization, yet which affect its capacity to be compliant (e.g. a bank has no real control over economical disturbances). Differently, internal factors are (to some extent) controlled by an organization (e.g. a bank's control over its credit risk approval procedures). Lastly, change related factors can be external or internal in nature either responding or driving the change (e.g. a change in Basel II legislation or adoption of a new credit product by a bank). Organizations manage these risks by adoption of a compliance program involving the different aspects of compliance discussed in the preceding sections.

With regard to risk identification and representation, organizations typically explore where risks lie within their business processes. An example in this area is [NS08], which argues that although compliance regulations pose many requirements, they fail to provide any specific recommendation on how an organization has to achieve compliance. As such, organizations typically develop ad-hoc solutions to achieve compliance of business processes to internal controls. To remedy this situation [NS08] suggests formalization of internal controls and how they relate to operational processes. Specifically, it considers accounts, risks, business processes and a set of controls as the main components of its formal model. Accounts form the high-risk items on an organization's balance sheet such as inventory. Processes relate to these accounts like purchasing and warehousing for the inventory account. In these processes there are risks that need to be managed, for example discrepancies in inventory levels. To prevent such risks controls (e.g. an inventory check) are imposed on the processes.

Another relevant work is [MR05], which proposes extensions to process models to express risks during design. Concretely, the paper presents graphical notations to cater for description of risk structure, goal effect, and dynamic risk inter-relationships. It also provides a taxonomy of process relevant risks, including goal, structural, data, technology and organizational risk. [MR05] and [NS08] essentially build on the work done on risk management in the context of business processes. Other examples include the work in [PGS+06] which defines an event calculus based technique for integrating risks in event driven business processes; and [GV06] which presents a framework with which time related conditions in compliance regulations can be verified against business processes using a formal logic.

## 2.6. Conclusions

The discussion in this section illustrates that a variety of compliance frameworks have been developed that address one (or more) aspects of compliance; being compliance discovery, specification, reporting, improvement and risk management. An interesting aspect of the current approaches to compliance specification and reporting is that almost all of them are reactive in nature. The aim of these proposed solutions is to provide the ability to check a

posterior whether business process designs are compliant. Reporting can then be done in a prospective or retrospective manner (i.e. prior or posterior to process execution). However, no facilities are offered for the case in which business process designs are not compliant. It would be interesting to investigate whether a more pro-active attitude can be employed. That is, rather than reactively determining whether a design is compliant by contrasting it with a set of compliance requirements, these requirements could also be used to generate a compliant design at runtime. This is particularly of interest given the fact that the applicability of compliance requirements is often dependent on runtime conditions. As such, what constitutes compliant or non-compliant behaviour is determined by runtime circumstances. This requires dynamic definition of process models, as e.g. touched upon in [GV06] (where obligations and permissions are used to generate control flow based work flow models).

The previous is related to another gap in existing work, which is the lack of attention for compliance monitoring and enforcement. Prospective reporting is typically performed to check prior to runtime whether or not a business design is compliant, whereas retrospective reporting is carried out to verify the execution behaviour of the process after its completion. However, as noted in the previous paragraph compliance is often runtime dependent. But this can not be adequately catered for with existing solutions. Also, it is not feasible with current solutions to take the results of compliance-related activities in a business process into account and the effects thereof on the behaviour of the process. As a simple example, failure of authorization by a customer will likely lead to abortion of an ordering process; as such disturbing the normal process flow. Additionally, authorization failure may prompt execution of additional activities to notify the security manager in real-time to create awareness for a potential security threat (e.g. in case a customer attempts to place a substantial securities requisition order). Cases such as these require that it must be possible to define responses to compliance successes as well as violations, which have the potential effect of altering the business process. Combined with the point made in the previous paragraph this also suggests a more pro-active approach to compliance specification and enforcement.

Finally, the discussed compliance proposals (like [GK07] and [LMX07]) are conceptually similar in nature in that they advocate: 1) the need for models to describe business processes including compliance-specific issues such as retention; 2) the need for models (i.e. policies and rules) to describe the compliance requirements to which business processes must conform; and 3) the explicit separation of these two types of model while at the same time establishing how they are related. Point 2 is especially relevant as the externalization of compliance requirements allows linking these requirements to particular compliance regulations. Furthermore, it makes them available for review and inspection. Moreover, often compliance requirements are only applicable contingent on particular circumstances. By making them explicit such contingencies can be expressed and taken into account when assuring compliance of business processes. Potentially they can even be employed at runtime to ensure enforcement of runtime specific compliance requirements. All these factors contribute to the aspect of compliance improvement. Point 3 is also of importance in that regard, since it allows the relation between a business process and its compliance requirements to be scrutinized for deviations.

### 3. Compliance discovery

In the previous section the notion of compliance program was introduced and several aspects of such program were explored. One of these aspects is concerned with compliance discovery, that is, finding out which compliance legislations are of importance for organizations and what the consequences are for the specification of their automated business processes. Though solutions for compliance discovery were already explored in Chapter 2, the topic of which compliance legislations are of importance (and the implications thereof for automated business processes) was not addressed. In this section this is remedied by studying several compliance legislations, in particular in the context of the COMPAS project. Concretely, legislations from different countries and continents are considered, since many international organizations have to assure compliance to legislations from various places (like North America and Europe). Moreover, the focus is on covering a diverse range of key compliance legislations in order to distil the types of requirement that may be involved. Also, the presented discussion includes both principle-based and rule-based compliance regulations. Principle based compliance regulations describe principles that an organization needs to adhere to without specifying exactly which requirements to meet or how this is to be done. In contrast, rule-based regulations give exact details on which compliance requirements to meet and how. As a final remark it should be noted that the presented analysis is not intended to be exhaustive. Rather, it serves to examine which compliance concerns are addressed in existing compliance legislations, and which types of business process are affected.

Having said that, in line with the above and the official Description of Work [DOW] in this section Basel II (<http://www.bis.org/publ/bcbs128.pdf>) on banking regulations, FINRA (<http://www.finra.org>) on securities trading, HIPAA (<http://www.hhs.gov/ocr/fedreg.zip>) on health care information handling, IFRS (<http://www.pwc.com/ifrs>), MIFID (<http://eur-lex.europa.eu/LexUriServ/site/en/consleg/2004/L/02004L0039-20060428-en.pdf>) on securities trading, the Sarbanes-Oxley Act (or Sarbanes-Oxley for short) ([http://frwebgate.access.gpo.gov/cgi-bin/getdoc.cgi?dbname=107\\_cong\\_bills&docid=f:h3763enr.tst.pdf](http://frwebgate.access.gpo.gov/cgi-bin/getdoc.cgi?dbname=107_cong_bills&docid=f:h3763enr.tst.pdf)) and Tabaksblat (<http://www.commissiecorporategovernance.nl/Definitieve%20code>) on corporate governance are analysed. It would go beyond the scope of this deliverable to discuss these legislations in detail. Rather, in the following we first give an overview of the different kinds of compliance requirements imposed by compliance legislations in section 3.1. Then, in section 3.2 we investigate the types of business processes affected by the legislations in order to determine their sphere of influence.

### 3.1. Types Of Compliance Concerns

As said in the introduction of this section one dimension of the analysis of compliance legislations constitutes determining the different types of compliance requirements they pose. By and large most legislation tends to address similar concerns, of which Table 1 below provides an illustrative overview.

	Basel II	HIPAA	IFRS	FINRA	MIFID	Sarbanes-Oxley	Tabaksblat
<b>Control flow</b>			X	X		X	X
<b>Information</b>	X	X	X	X	X	X	X
<b>Locative</b>	X	X				X	X
<b>Resource</b>						X	X
<b>Temporal</b>	X	X				X	X
<b>Monitoring</b>	X	X				X	X
<b>Payment</b>					X		
<b>Privacy</b>		X			X		
<b>Quality</b>	X	X	X	X	X	X	X
<b>Retention</b>	X	X	X	X	X	X	X
<b>Security</b>		X				X	
<b>Transaction</b>			X		X	X	X

**Table 1: Compliance Legislation by Compliance Requirement Type**

As can be seen in Table 1 two groups of compliance concern are addressed by compliance legislations. The first group constitutes so-called basic compliance concerns. These concerns are basic in the sense that they pertain to the basic structure of business processes. They are five fold:

- The **control flow** compliance concern encompasses requirements concerning how things are done in business processes (i.e. what activities are carried out and in what order). For example, the International Financial Reporting Standards (IFRS) statements target the financial reporting processes of organizations, mandating among others requirements for what activities to perform in the process control flow.
- The **locative** compliance concern addresses requirements concerning the location where business process activities are carried out. For example, the Sarbanes-Oxley Act's interpretation in the IT Control Objectives for SOX by the Information Systems Audit and Control Association (ISACA, <http://www.isaca.org/sox/>) identifies as one important aspect the location at which information is stored.
- The **information** compliance concern deals with the information used and produced in business processes as well as the syntax and semantics of this information. Chapter I of the MIFID regulation provides several articles for example for when dealing with new clients, e.g. with regard to the provisioning of high quality information (both about the firm and particular services and products). Another example is Basel II, which places demands on the information stored during credit provisioning in order to facilitate future credit risk assessment.

- The **resource** compliance concern considers the question of which resources are used within business processes (e.g. employees and customers, and computerized systems). To illustrate, the 'Code Tabaksblat' promotes the usage of separation of duty (just as other legislations such as SOX).
- The **temporal** compliance concern takes requirements concerning when things are done/must not be done within a business process into account (e.g. in terms of relevant business events). To exemplify, Pillar II of the Basel II accord seeks to enforce monitoring capabilities for banks (as well as supervisors) to assess their capital adequacy. This requires definition of proper events in order to capture temporal occurrences.

As Table 1 shows, some legislation (such as Sarbanes-Oxley and Tabaksblatt) covers all five basic compliance concerns. Others, for example Basel II, only stipulate information, locative, and temporal concern related requirements.

The second group of compliance concerns in Table 1 is more advanced in nature. Advanced compliance concerns build on basic ones, that is, without the latter advanced compliance concerns can not be addressed. Seven advanced concerns are often found in compliance legislations:

- The **monitoring** compliance concern deals with compliance requirements that organizations have to meet with regard to the (real time) awareness of the state of a business process; as such building on the temporal concern. Basel II, mentioned for the temporal compliance concern, requires adequate monitoring of credit provision processes. This suggests the presence of monitoring mechanisms inside these processes to warn when provision of credit endangers capital levels
- The **payment** compliance concern recognizes the need of organizations to specify payment related requirements, e.g. to comply to MIFID regulations concerning the payment of provided financial services. It builds on the control flow concern in the sense that payment specifics are typically associated with the activities performed within a business process. An example is that payment must be done electronically and such that it can not be repudiated.
- The **privacy** compliance concern encompasses compliance requirements related to privacy in business processes. It builds on both the information and resource concern. In relation to the information concern, the privacy compliance concern poses requirements regarding the sharing and usage of personal information within a business process. In context of the resource concern, the privacy compliance concern comprises constraints regarding the anonymity of involved process participants (e.g. A customer or employee). For example, the Health Insurance Portability and Accountability Act (HIPAA, <http://www.hhs.gov/ocr/fedreg.zip>) is aimed at the medical domain. It defines a Privacy Rule which mandates the capability to indicate which information is private and which may be disclosed (to whom), how consent may be acquired, and how disputes are resolved.
- The **quality** compliance concern is relevant for covering preferences and demands concerning the level of quality in business processes. It builds on the control flow and information concern. With regard to the control flow concern quality comprises issues such as responsiveness, reliability, accessibility and availability of business process activities (and their automated implementations). Differently, in relation to the information concern quality constitutes the accuracy and completeness of information, e.g. of importance for financial information. Examples of quality requirements are

found throughout many regulations, e.g. relevance and reliability of financial information (IFRS), accuracy of personal health information (HIPAA), and completeness of data while calculating the credit risk (Basel II).

- The **retention** concern revolves around the proper archiving, retrieval and disposal of information. It builds on both the information concern (regarding the storage of information) as well as the temporal concern (regarding the logging of events). Requirements in the retention concern encompass issues such as the period of storage, location, method of disposal and etceteras for process documents and/or events. To illustrate, retention of records is covered in section 8, Chapter III of MIFID detailing that transaction related documents must be kept a minimum of 5 years and client relation details for the length of this relation.
- The **security** compliance concern addresses security related compliance requirements for business processes. It builds on the control flow and information concern. For control flow the security compliance concern is of importance for ensuring authenticated and authorized access to resources within business processes. Relating to the information concern, it deals with preserving the confidentiality of information during message exchanges as well as detection of information tampering. These kinds of requirement are found in e.g. the HIPAA Security Rule, which addresses a wide range of security requirements encompassing issues surrounding authorization, authentication, integrity and confidentiality.
- The **transaction** compliance concern involves transactional requirements for business processes, such as that a group of related securities (e.g. investment bonds, stocks) orders from a customer must be placed and effectuated in an all-or-nothing manner. For example, to ensure accuracy of financial information IFRS poses transaction demands to assure that failed financial transactions do not lead to inconsistent and incorrect financial information

Interesting in this regard is that the dominant concerns in these legislations constitute quality and retention, followed closely by monitoring and transaction. Relatively less widespread are privacy, security and payment. Also, it is important to note that advanced compliance concerns are typically cross-cutting over multiple basic concerns. To illustrate, retention of information usually not only refers to exchanged messages (part of the information concern), but also monitored events (part of the temporal concern). This implies that the specification of advanced compliance concerns must also be cross-cutting in nature; and thus that the integration of such specification details into business process designs requires an approach that is orthogonal to the solutions used to capture the basic concerns. Moreover, it is crucial to observe that advanced compliance concerns can be intertwined as well. For example, in compliance legislations such as Sarbanes-Oxley it is not only stated that certain information must be retained, but also that this must be done in a tamper proof manner. That is, information must be stored such that its authenticity can be proven if so required. This suggests a mixture of security requirements applied to retention related requirements in order to assure compliance within a business process.

### 3.2. Types Of Affected Business Processes

Section 3.1 discussed compliance legislations from the point of view of the type of compliance concern they express. A different perspective on compliance and business processes is shown in Table 2, which gives an overview of example processes affected by the considered compliance legislations.

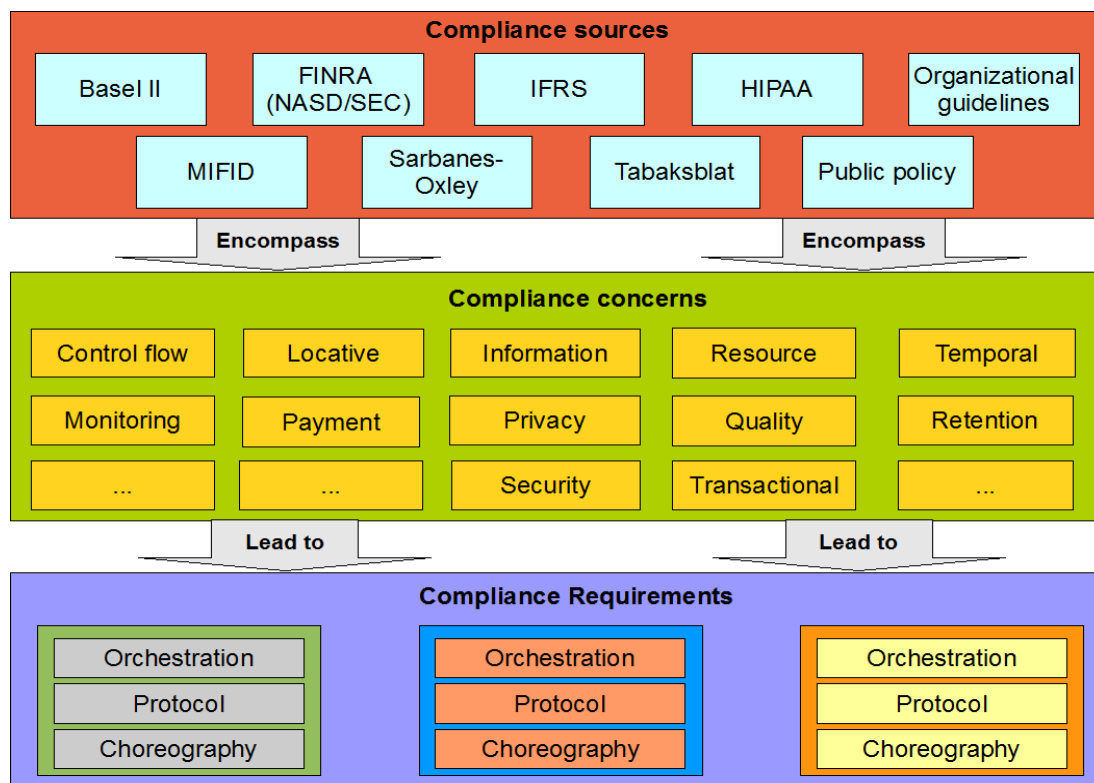
	<b>Orchestrations</b>	<b>Protocols</b>	<b>Choreographies</b>
<b>Basel II</b>	credit risk assessments, credit provisioning	disclosure of information	sharing of risk related information
<b>HIPAA</b>	Insurance eligibility assessment	client enrollment, client access protocols	health information communications
<b>IFRS</b>	financial reports development	financial reporting processes	financial reporting processes
<b>FINRA</b>	trading of securities	sales and marketing procedures, provision of transaction information	sales and marketing procedures, provision of transaction information
<b>MIFID</b>	order execution, conflict handling procedures	order processing, client enrollment, provision of transactional information	order processing, client enrollment, provision of transactional information
<b>Sarbanes-Oxley</b>	financial reporting processes, sales, acquisition, inventory, tax	provision of financial information, provision of internal control management	provision of financial information, provision of internal control management
<b>Tabaksblatt</b>	financial processes, commissioner appointment, end-of-contract reward calculation	interaction with shareholders, sharing of financial information	interaction with shareholders, sharing of financial information

**Table 2: Impact of Compliance Legislation on Business Processes**

The table shows for example that Basel II affects credit risk assessment procedures, disclosure of risk information and sharing of such information. Differently, Sarbanes-Oxley impacts all financial reporting processes, but also provision of financial information to external parties. An important distinction made in this regard in Table 2 is the fact that compliance legislations influence different types of business process. Specifically, such legislations can have consequences for private (internal) processes, exposed protocols as well as public processes. In relation to SOA enabled processes these are typically referred to as orchestrations, protocols and choreographies. To illustrate, in order for a bank to be Basel II compliant it must ensure that its private credit risk assessment process meets stipulated requirements. At the same time though the sharing of risk related information with other parties, a public process, must also transpire in conformance with applicable regulations. As such, ensuring compliance of organizational processes is a wide scale undertaking, which potentially crosses organizational boundaries; in turn affecting the manner in which organizations interact with other parties. This is important to realize as specialized specification methodologies, methods, techniques may be required to account for the fact that internal processes are controlled by individual organizations; while public processes are only partially under their control. An alternative is that the developed specification solutions can be applied in context of these different types of process in an uniform fashion.

### 3.3. Conclusions

In the previous we briefly surveyed a sub-set of compliance legislations to analyse their scope in terms of different compliance concerns and affected business processes. Although the survey of compliance legislations is at best illustrative in nature (particularly given the fact that many non-regulatory compliance sources exist), the analysis contributes to compliance discovery in the sense that the identified concerns are generic in nature; and as such provide a faithful indication concerning what types of requirement need to be addressed when dealing with compliance. They also reflect the variety of business processes that can be affected by compliance legislations ranging from private to public processes. This is visualized in Figure 2 below.



**Figure 2: Legislative Compliance Concerns for Business Processes**

Naturally, whether or not the identified compliance concerns affect service enabled business processes depends on whether or not these processes (or parts thereof) are automated by employing service oriented computing technologies during their execution. In the event that this is the case the concerns must be addressed and the question as to how to enforce internal controls within these automated processes becomes important. This in turn puts the focus on the modelling of such concerns in business process designs, which is an important aspect of compliance specification. Compliance specification within business processes is the topic of section 4.

## 4. Compliance specification

In the previous section a large body of compliance legislations was examined in the context of compliance discovery resulting in a set of different kinds of compliance requirement. In this section current work in the area of compliance requirement specification within business processes is surveyed in order to assess to what extent they can provide support for the modelling of these different requirement types. For structure purposes the discussion is divided in two parts. In section 4.1 languages for modelling the basic compliance concerns for business processes are examined. Next, languages for modelling the advanced compliance concerns are investigated in section 4.2.

### 4.1. Modelling basic compliance concerns

As said, basic compliance concerns represent the types of compliance requirements applicable to the basic structure of business processes. To recall, five of such concerns were identified in the analysis of compliance legislations, being the control flow, information, locative, resource and temporal concern. Business process modelling has a long tradition and consequently several well established tracks for how to model these concerns have been explored. More recently, work has also been done on how to capture more advanced characteristics alongside with the basic business process structure. In the following we discuss the main ideas developed in industry and academia in these areas. We also look at the available tooling for business process modelling.

#### 4.1.1. Business process modelling approaches from industry

Current approaches in industry mostly focus on the modelling of business processes using the SOA paradigm, in which process definitions are expressed as collections of orchestrated services. Additionally, much attention has been (and remains to be) on the usage of workflows to describe business processes. Finally, a strand of more abstract and generic business process modelling solutions exists.

##### 4.1.1.1. SOA and workflow based business process modelling

Starting in the context of service enabled business processes, the de facto standard that has emerged for SOA (web service) based process modelling is Web Service Business Process Execution Language (WS-BPEL). WS-BPEL is an XML based language in which a process is viewed as a set of complex business interactions involving multiple parties, where each business interaction is perceived as the exchange of a series of messages between these parties. Each process model consists of four parts: the containers, partners, fault handlers and process section. The containers section defines the data containers that are used by the process to maintain state data during execution. These can thus be used to address information requirements. The parties involved in the business process are defined in the partners section covering resource and partially locative requirements. Exception handling behaviour is defined in the fault handler section to some extent addressing temporal requirements. Lastly, the process section contains the definition of the behaviour of the business process describing its control flow requirements.

Two kinds of business processes are distinguished in WS-BPEL: abstract and executable processes. Abstract processes describe business interactions by precisely specifying the message exchange behaviour of the parties involved without revealing their internal implementation. Importantly, there is a separation from the public and private parts of the

business process. Executable business processes are similar to their abstract counterparts in the sense that they also provide a specification of the message exchange behaviour of the parties involved in the business process. However, in an executable process the external aspects of the process are not separated from its internal workings. Focusing on public processes only, relevant works include for example the Web Service Conversation Language (WSCL, <http://www.w3.org/TR/wscl10/>) and Web Service Choreography Description Language (WS-CDL, <http://www.w3.org/TR/ws-cdl-10/>). These XML based languages focus on specifying the coordination of a set of web services from a global point of view (rather than from an individual organization's viewpoint as is the case with orchestration). WSCL covers control flow, information, resource and to some extent locative requirements, while in addition WS-CDL also addresses temporal requirements in relation to the occurrence of faults.

Another avenue for business process modelling that has been pursued within industry, is the area of workflow. Workflow is concerned with the automation of procedures where documents, information or tasks are passed between participants according to a defined set of rules to achieve, or contribute to, an overall business goal (<http://www.wfmc.org/standards/docs/tc003v11.pdf>). Workflow initially focused on intra-organizational processes assuming a single model and centralized execution engine. A noteworthy example in this regard is XPDL, short for XML Process Definition Language ([http://www.wfmc.org/standards/docs/TC-1025\\_xpdl\\_2\\_2005-10-03.pdf](http://www.wfmc.org/standards/docs/TC-1025_xpdl_2_2005-10-03.pdf)), which provides constructs for describing processes in terms of directed graphs of tasks (control flow requirements) fulfilled by roles (resource requirements) using data (information requirements). However, for the last decade or so distributed workflows have been receiving growing attention. One example is found in [CH01] where the idea is to create a virtual cooperative process across organizations. Once the design is accomplished, this process is chopped into individual pieces that are executed by the respective individual organizations. Many ideas developed within workflow have found their way into the current industry standards for web service based process modelling such as the already discussed WS-BPEL.

#### **4.1.1.2. Generic business process modelling approaches**

At the same time as work on WS-BPEL and related standards was under way, a set of more generic business process modelling proposals were also being developed. One such solution is the already mentioned Business Process Modeling Notation, developed by the Business Process Modeling Initiative (BPMI). This language provides an abstract model for expressing business processes and supporting entities. The language can be used for expressing abstract and executable processes, which address aspects of enterprise business processes. These aspects include activities of varying complexity (control flow requirements), transactions and their compensation, data management (information requirements), concurrency, exception handling (temporal requirements) and operational semantics (which are currently underspecified). Swimming lanes are used to convey the notion of partners within a process (either belonging to the same or different organizations).

Interesting work done by the ebXML consortium is the Business Process Specification Schema (BPSS, <http://www.ebxml.org/specs/ebBPSS.pdf>). This schema provides a standard framework with which business systems may be configured to support the execution of business collaborations consisting of business transactions. Business transactions are realized through the exchange of business documents (informational requirements) between the requesting and the responding party. Collaborations can be binary or multi-party in nature. In a binary collaboration exactly two business partners (locative and resource requirements) are

involved. It is expressed as a set of business activities (control flow requirements) choreographed in a particular manner. These activities help progress the state of the business process (temporal requirements). Additionally, the ebXML consortium provides the Collaboration Protocol Profile (CPP) and Collaboration Protocol Agreement (CPA, [http://www.oasisopen.org/committees/ebxml-cppa/documents/ebCPP-2\\_0.pdf](http://www.oasisopen.org/committees/ebxml-cppa/documents/ebCPP-2_0.pdf)) with which organizations can model their cooperative capabilities as well as specify negotiated agreements, both of which are defined in the same terms as BPSS. BPSS also supports specification of some security and transaction related specifics.

Even more general in nature, another option is to utilise generic modelling approaches to describe business processes. A prime example of this is UML [BRJ98], which provides a standardised visual specification language for object oriented modelling. UML offers several types of diagram that can be used in conjunction to model business processes. For example, class diagrams can be defined to express the information requirements of processes concerning what information objects are involved and how these are related. Activity diagrams can then be specified to capture the process' control flow requirements by depicting what activities are performed, in what order and under what conditions. These activities are triggered by events as well as generate new events, which can be described using state chart diagrams (temporal requirements). The activities can be linked to class diagram objects as operations performed on these objects. Use case diagrams can be specified to capture the interaction between a business process and the actors involved (resource requirements). Use case diagrams are very abstract in nature though and as such do not naturally lend themselves for associating actors with concrete activities; something which is necessary for compliance with regard to allocation of duties. Lastly, support for the modelling of locative requirements is lacking in UML. Alternatively, UML has also been employed to develop UML notations for e.g. WS-BPEL by utilizing its class diagram to capture the concepts in BPEL and how they relate to one another.

#### **4.1.2. Business process modelling approaches from academia**

Different from the work done in industry (which tends to gravitate round the control flow dimension of business processes), the focus of academia has been more diversified when it comes to business process modelling. A branch that is worth mentioning from the scientific community has explored the usage of so-called event-driven process chains (EPCs). The emphasis in such approaches is on the definition of the control flow of the business process in terms of events and functions (temporal and control flow requirements). Functions perform some business activity when they are triggered by events. Subsequently, they produce events as they carry out those activities. As such, the control flow is expressed as a sequence of alternating events and functions; which from a compliance monitoring point of view is interesting. An XML based example of EPCs is the EPC Markup Language (EP-ML) [MN04]. In addition to the specification of the process control flow other kinds of information may be included in an EPC. Such information usually depicts the use of resources for carrying out of functions (resource requirements) and the interactions with the data structure (information requirements) of the organization. Because of their inherent distributed nature EPCs are suitable for both intra-organizational and inter-organizational processes. Moreover, given the importance of monitoring in compliance, events are a key component of business processes. It is unclear though how more advanced concerns can be modelled in EPCs.

Another interesting research area is that of role and agent based approaches. The motivation behind such approaches is that often in process models it is unclear who is responsible for what (as the emphasis tends to be on decomposition related to function). Therefore, works

like found at (<http://www.ebpml.org>) and [JFN+96] emphasize definition of resource requirements. The work at (<http://www.ebpml.org>) for example suggests the foundation of a meta-model of a business process definition in terms of message exchanges (information requirements) between two roles (resource requirements). This enables the definition of decentralized processes which involve business-to-business collaborations as well as user interactions and application-to-application integration in a technology neutral way. [JFN+96] presents an autonomous agent based approach to business process modelling management in which processes are viewed as interactions among independent agents. Differently, work in the semantic web has concentrated on the informational aspects of business processes taking the idea that information is vital to the business process. Such approaches start out by defining the data structure in a process (information requirements). Subsequently, operations that can be applied to this structure are developed, which constitute the different process tasks (control flow requirements). Other constructs can then be added if desired. A well known exponent of such form of business process modelling is DAML-S, which is being developed by the DAML Program (<http://www.daml.org/services/daml-s/0.9/>).

A separate category of process modelling languages is formed by more formally oriented works. Formal techniques contribute to the formal verifiability of business process models (e.g. concerning deadlock). However, their usage requires expert knowledge. As such, a combination of a user friendly modelling language and an underlying formalization would be a good solution. One example is simple finite-automata, with which processes are described as devices that maintain the state of something at a certain time and can alter this state in reaction to input as well as cause an action or output as a result of a changing state. Petri nets offer another graphical technique, and are a special form of graphs constituting of places, transitions, directed arcs and tokens. Places are connected via directed arcs to transitions and vice versa. Places contain tokens, which may represent signals, events, conditions, and so on. Transitions are fired through the presence of tokens in their in-place(s). As a result the distribution of tokens is changed. Example work includes [HB03] using Petri Nets for workflow and service composition representation respectively.

More symbolic in nature are the techniques of formal logics and process algebras. Formal logics define processes as collections of predicates based upon which reasoning can take place to analyse characteristics of these processes. Logic based approaches are discussed in detail in section 5. Process algebra is a formal description technique designed for complex computer systems, especially those involving communicating, concurrently executing components [BPS01]. An algebra that has received some popularity, specifically in the context of web service based business processes, is pi-calculus [MIL93]. In pi-calculus a process can be described as a collection of interacting services over communication channels. A last approach worth mentioning is REO [ARB04]. REO is a constraint automata based approach for formal definition of foundational models for coordination and composition. It is characterized by the fact that they can cater for the description of the behaviour of active entities that (1) are fully compositional, and (2) can express arbitrary mixes of synchronous and asynchronous behaviour. Moreover, recent work is under way to extend the basic automata with support for verification of compliance related concerns.

### **4.1.3. Advanced business process modelling approaches**

In addition to the modelling of the basic structure of business processes, more recently the integration of more advanced concerns has received increasing interest. Notable in this regard is the approach described in [CM07], which proposes a so-called aspect oriented approach for business process modelling with WS-BPEL. The central idea is that while the process model

is prescriptive in nature, aspects are attached to them expressing different kinds of more advanced requirement in a declarative manner. This is done by attaching aspects to basic process model constructs (like activities). When the business process is to be executed, the basic protocol is expanded to include activities that enforce these aspects. To customize the point at which this is done, point-cuts express conditions on the business process runtime circumstances. If the conditions are met, then the so-called advice is carried out (which constitutes the actual activity to be performed). [BVJ+06], [BJP+07] and [BJ07] present a similar approach as [CM07], but their approach is more conceptual in nature and utilises the more expressive Prolog for point-cut definition.

The utilization of such approach in relation to integration of compliance requirements into business processes has been explored in for example [SCM+07], which proposes an aspect-oriented based approach for linking compliance to business protocols (i.e. abstract business processes). Another relevant work is [SEH02], which develops an extensive classification of non-functional requirements for web service specification. This classification covers diverse areas such as quality, security, payment, location, and trust. Although not directly applied in the context of compliance and business process modelling, the identified properties in [SEH02] (and the work on the corresponding web site at <http://www.service-description.com>) are useful as a starting point for the description of (some of the) advanced compliance requirements identified in section 3. Also, interestingly the developed property models can be combined and linked, e.g. to relate security related statements to specific locations and dates.

#### 4.1.4. Tool support

From a practical point of view a variety of both commercial and open source tools for business process modelling (and/or execution) can be found, which are mostly based on industry developed standards. For WS-BPEL a multitude of orchestration engines have been developed including the Oracle BPEL Process Manager, the IBM WebSphere Business Integration Server Foundation, the Active Endpoints ActiveWebflow Server and the open source ActiveBPEL engine. These engines are all Java based implementations of WS-BPEL. For .Net implementations also exist; e.g. Microsoft BizTalk 2004 and OpenStorm Service Orchestrator. Design tools are available as well such as the Oracle BPEL Designer, the Netbeans BPEL Designer component, the IBM Websphere Integration Editor and the Active Endpoints ActiveWebflow Designer. Open source engines are also available, such as the Apache Orchestration Director Engine (<http://ode.apache.org/>) and the Eclipse BPEL Designer (<http://www.eclipse.org/bpel/>). For BPMN several design tools have also been developed ([http://www.bpmn.org/BPMN\\_Supporters.htm](http://www.bpmn.org/BPMN_Supporters.htm)). To name a few, these include implementations from Borland Together, Cordys, Fujitsu, IDS-Scheer (ARIS), Intalio, Pega Systems, Sun and Visual Paradigm. Sometimes BPMN models can then be transformed into BPEL equivalent processes for execution purposes; where BPMN is used as graphical notation. This is not always feasible though as e.g. discussed in [ODH+07]). XPDL has several implementations as well (<http://www.wfmc.org/standards/xpdl.htm>), such as those by the Fujitsu Interstage Business Process Manager, the BEA Systems AquaLogic Enterprise Repository and BPM Suite, the IDS Scheer Business Architect and the TIBCO iProcess Suite. ebBPSS also has its share of tools (<http://www.ebxml.org/tools/index.htm>), for example the Business Process Integrator Suite from Sybase, the ebMS from Briyante, and the BusinessWare suite from Vitria. Finally, for UML a wide range of tools are available ([http://en.wikipedia.org/wiki/List\\_of\\_UML\\_tools](http://en.wikipedia.org/wiki/List_of_UML_tools)) such as Borland Together, IBM Rational Rose, Microsoft Visio and Netbeans 5.5 Enterprise Pack.

In conclusion, there is a wide array of business process modelling solutions currently available. Some, like WS-BPEL are technological in nature describing business processes in terms of services interacting through message exchanges. Others, such as BPMN, are higher level in nature utilizing concepts more familiar to business people to describe processes. Ultimately, both types of language will be needed as specification of business processes requires intuition while their execution requires a high level of detail. From a compliance point of view current business process modelling solutions will likely have to be extended. These solutions are typically adequate for addressing control flow, resource and information requirements, however are lacking somewhat in the area of temporal and locative requirements. Also, it must be carefully considered that the chosen languages are either extensible in nature with regard to modelling of more advanced compliance concerns; or that they accommodate an approach as suggested by [CM07]. As such, appropriate extensions of existing solutions or definitions of new languages may have to be developed or somehow accommodated for within current solutions.

## **4.2. Modelling advanced compliance concerns**

The analysis in the previous section reviewed current solutions for capturing the basic structure of business processes. These solutions typically cover several of the five basic compliance concerns. However, compliance legislations also impose requirements going beyond the basic concerns. Particularly, in section 3 seven advanced concerns were found to be of importance for the discussed legislative bodies, being monitoring, payment, privacy, quality retention, and security. In the following these are briefly reiterated, after which relevant works for modelling requirements in those areas are investigated. For each concern the focus is on extrapolating important issues and after which they are analysed to assess whether they can help capture such requirements in the context of compliance and business process modelling.

In this regard it is also considered whether these constructs allow the compliance requirements to be captured at different levels of abstraction. The reason is that, as we observed in section 3, compliance legislations are often abstract in nature only specifying high level compliance objectives. However, for the actual enforcement of such abstract objectives concrete and unambiguous control requirements are required; particularly also with regard to identifying the technological solutions employed. Thus, in order to be able to specify compliance requirements in sufficient detail and at the same time be able to trace these requirements back to abstract statements, it is necessary to consider compliance concerns at multiple abstraction levels; and moreover make explicit how these are related to one another. This in turn will enable organizations to refine their business oriented compliance requirements into technologically oriented ones; establishing a clear path between these requirements that can be used by organizations to facilitate communication and reasoning about compliance among their different compliance-invested stake holders.

### **4.2.1. Monitoring**

The monitoring compliance concern constitutes compliance requirements for observing the state of business processes by observing the events that they generate. Because of compliance legislation for example they must be capable of stipulating that high risk events must be observed and analysed, e.g. for a bank that if the credit balance drops below a certain threshold, this must immediately be brought to the attention of the bank manager. Another example is that organizations will want to monitor their activities in order to guard progress, such as while processing an order from an important customer. Moreover, the enforcement of

compliance requirements should be monitorable as well, so that in case of violation the organization can react in a promptly manner. For example, if authorization for an activity fails, then the resulting 'authentication failure' event should be monitored so a proper response can take place. This requires definition of noticeable events as well as identifying who publishes and is required to receive them, how they should be handled and so on.

#### 4.2.1.1. Definition

One area that has been explored concerns the conceptualization and consequent definition and specification of events. Although works differ sometimes in the exact notation used, they tend to share the same informal notion that events can be considered to be occurrences of some sort, which are of relevance for the organization. For example, in relation to computerized systems [LUC02] defines an event as an object that can be subjected to computer processing. Such object comprises of data such as the date and time of occurrence (called a timestamp), its location (that is, the point from which the event originated), its source (i.e. the entity who caused the event), and its severity (indicating the seriousness of the event, e.g. a fault message, logging message, and etceteras). Events also usually have an identifier allowing them to be uniquely identified. These data components are often referred to as event attributes for abbreviation purposes. Similar attributes are also found at (<http://www.ibm.com/developerworks/library/ac-cbe1/>), which defines the Common Base Event (CBE) data model to describe events published by software components in a standardised manner. To this end the language prescribes the format of events as well as the content of their description expressed as properties.

Identified properties in CBE include: 1) a local and global identifier; where the former is an identifier provided by the source component of the event, while the latter is assigned by an event processing facility. These might be useful to distinguish between events in relation to a single business process or in relation to multiple organizational processes; 2) the time of creation of the event, i.e. a time stamp; 3) the severity and priority of the event, where the latter expresses the level of urgency associated with dealing with the event; 4) the reporter and source component allowing to distinguish between the source of an event and the entity actually publishing it; 5) the situation in which the event occurs, e.g. starting or stopping a component. In relation to monitoring this can be useful to group events as well as treat events differently based on the type of situation in which they occur; and 6) a repeat counter to reflect the number of times an event occurred, optionally within a specified time period.

CBE has been adapted to the context of SOA in the Web Services Distributed Management (WSDM) specification (<http://docs.oasis-open.org/wsdm/wsdm-muws1-1.1-spec-os-01.pdf>, <http://docs.oasis-open.org/wsdm/wsdm-muws2-1.1-spec-os-01.pdf>). In WSDM event descriptions can be flexibly defined, and can have at least a situation element, sequence numbers and event correlation statements, and message texts for providing information on the context of the event. They also have an identifier, and source and reporter component tags. WSDM builds on the Web Service Topics specification (<http://docs.oasis-open.org/wsn/wsnwsttopics-1.3-spec-os.pdf>), which allows definition of event topics in a standardised XML based format. WSDM consists of two specifications, which describe how to manage web services as well as how to use web services for management respectively.

The attributes discussed so far can be considered to be meta-data like properties that help process and reason about events. To signify their actual meaning events typically also contain data describing the activity they signify. In the context of SOA based business processes this data is usually comprised in messages; e.g. in WS-BPEL where the receipt and sending of messages implicitly constitute event occurrences. [LUC02] refers to this as the significance of

an event. The CBE captures this information in a so-called message data element and a user friendly representation of this message. These works also enable capturing the context in which events take place in relation to other events. In the CBE this is facilitated in two manners. One is through the inclusion of so-called context data elements, which help correlate events in an execution path. An additional option is by defining associated events, which allow for event grouping or definition of parent-child relationship. The first supports expression of causal relations while the second enables complex events to be formed out of other events,

[LUC02] describes similar relations among events comprising its relativity. Concretely, three forms of relationships are identified being time, causality and aggregation. Along the dimension of time the relation among events is determined by their timestamps; where an important consideration is whether event timestamps were generated by the same computer clock or by different clocks. If events arrive in order conform the used computer clocks, then it is possible to employ event stream processing (ESP). ESP essentially consists of a set of technologies, techniques, algorithms and tools that take advantage of the fact that events arrive in order. This allows using processing algorithms that only have to remember events a very short period of time, enabling them to be processed in a rapid manner. As soon as the events have been processed and their effects determined, the events pass and can be discarded.

Due to differences in clocks (and whether or not these clocks have been synchronized) the meaningfulness of a time based ordering can be limited. This is important for the monitoring of SOA enabled business processes, since these are typically distributed in nature involving multiple clocks (which in addition can potentially span multiple time zones). Also, events do not always arrive in a neat order in a business process. Therefore, a causality based ordering is often used. Causality expresses a dependency relationship between events; e.g. that if event B depends on event A, then A caused B. Conversely, if neither A caused B (or vice versa) these events are independent from one another. Causality relations are useful as they allow to trace the behaviour of business processes independent of the time of occurrence, e.g. to find the origin of an exceptional event (which might be found in a completely different part of the business process). They are specified in [LUC02] as causal vectors in event definitions, where such vectors comprise a list of event identifiers. Based on causality events can then be processed using complex event processing (CEP) techniques.

Lastly, with aggregation events can be grouped to describe more complex events; where the complex event occurs because its contained sub-events have transpired. This form of grouping of crucial for monitoring as it enables organizations to monitor events on different levels of abstraction. [LUC02] refers to this as addressing the information gap existing among the abstraction level of the data provided by low level system events and that what is useful for the organization. To exemplify, the occurrence of a 'sales order pending for approval' event can be of importance for a sales process employee, but is not interesting by itself for the sales process manager. Rather, he/she will be interested in the outcome of sales orders likely comprising multiple events related to the sales order processing.

In this regard filtering through event patterns is an important tool. Event patterns allow capturing conditions on events (both their meta-data and signified activity information) in order to find and select particular events. [LUC02] describes how this for example enables detection of events made up of otherwise unrelated, independent events, grouping of events all occurring within a specific time frame (context-sensitive), and identification of sales order events related to orders above \$500 (content-sensitive). [LUC02] also defines constructs to group events (for example through (exclusive) disjunction, conjunction, and periodic and a-

periodic occurrence) in its rule based event pattern language called RAPIDE. These allow expression of statements such as that a high risk order event takes place only if the value of the order is above \$1500 and the customer has a bad credit history; where thus the aggregate event only transpires if specific sub events take place.

A language like RAPIDE and the expressive power that comes with it are crucial in order for organizations to be able to depict what business process occurrences are worth monitoring and logging. Otherwise, the sheer amount of events would be overwhelming and lead to information overload. Moreover, it enables the different stake holders in an organization to monitor exactly those events that they are interested in; where they furthermore have the possibility to drill down to see the underlying events that caused monitored events to occur (using the causal and aggregation relations existing among events).

#### **4.2.1.2. Detection**

Definition of events is crucial for monitoring, but by itself not sufficient. Rather, detection of events that are deemed important must be facilitated. The most straightforward is identification of primitive events, that is, events that are not composite in nature. Primitive events can be expected to be detected by systems, e.g. database events will be detected by code built in to read and write operations. Following this train of thought it seems logical to expect that in the context of business processes, the engine responsible for executing a process is capable of detecting events. However, given the discussion in section 4.1 and the found lack of support for a strong notion of events, it may be the case that actual detection will have to be done by an entity other than the business process engine (e.g. based on the messages received and send within the process).

This need not necessarily be a problem, as this has to be done in any case for more complex events; as these can not be directly detected within business processes. [LUC02] suggests an approach in which primitive events are fed to an event processor that employs RAPIDE based pattern rules to identify more complex events. Alternatively, if the process language has a first class notion of events, then detection can be done directly. This is for example the case for business processes defined and executed using event-process chains (like EP-ML described in section 4.1). Another option is to include explicit sensors in the process to ensure detection, e.g. done in relation to WS-BPEL in [JG06].

#### **4.2.1.3. Notification**

Once events have been detected, subsequently notification thereof to appropriate entities (both human and/or computerized) must take place in some form or another. Different kinds of event notification mechanisms can be utilised for this purpose. Web Services Base Notification ([http://docs.oasis-open.org/wsn/wsn-ws\\_basenotification-1.3-spec-os.pdf](http://docs.oasis-open.org/wsn/wsn-ws_basenotification-1.3-spec-os.pdf)) describes a publish-subscribe mechanism for web services in which one service produces events to which other services can subscribe/unsubscribe. Specifically, it provides an event independent notification framework in which a notification producer produces notifications, which are then received by notification consumers. Interestingly, the framework distinguishes between 'raw' events versus 'notify' events.

Usage of raw events allows quickly dispatching events to interested parties without intermediate processing. However, the disadvantage is that in a distributed business process this will likely lead to a non-uniform representation and thus handling of compliance related events. Moreover, it allows addition of content independent information about the event to be included in the notification, such as the meta-data elements discussed above. Consumers can

register to multiple notifications, where no order is imposed on how notifications occur. Optionally, a consumer can define filters to limit the amount of notifications it receives. This allows publishers to publish events relevant from their perspective (e.g. for logging), while it enables consumers to receive only those events of interest to them.

We also find the publish-subscribe mechanism in other works, e.g. Web Services Eventing (<http://www.w3.org/Submission/WS-Eventing/>), which defines the notion of delivery mode to capture different styles of notification. WS-Eventing currently defines only the publisher/subscriber mechanism (referred to as the push mechanism, since events are pushed to interested parties). It also includes the feature of capturing characteristics about the subscription like the period in which a subscriber wishes to receive events from a publisher. Push based notification will be a typical choice in the context of compliance and business process monitoring, given its distributed and often asynchronous nature of communication and execution.

Web Service Base Notification observes though that there are cases in which this is not desirable and/or feasible; e.g. if the consumer of the event is behind a firewall or unable/unwilling to provide the necessary service operations to allow notification. To this end Web Service Base Notification defines a pull style mechanism in which the notification producer offers a set of operations with which consumers can retrieve events. This allows consumers to receive notifications on their own terms, also e.g. with regard to whether they wish to receive events on a per-event basis or e.g. as a daily digest. In relation to the monitoring of business processes, one way in which to offer support for such pull mode is by extending business process models with the necessary constructs. This would mean though that affected services in the process would have to provide pull-related notifications operations.

A better option is to use a brokered notification framework, which facilitates both push and pull based notification; as e.g. described in Web Services Brokered Notification ([http://docs.oasis-open.org/wsn/wsn-ws\\_brokered\\_notification-1.3-spec-os.pdf](http://docs.oasis-open.org/wsn/wsn-ws_brokered_notification-1.3-spec-os.pdf)). The basic idea is that there is an entity functioning as an intermediate between publishers and consumers; in the context of compliance and business processes between a business process and those monitoring the process. Such entity is typically referred to as a broker. This broker implements the functionality to allow publishers to publish events with the broker and subscribers to subscribe to events (conform to a particular notification mode). Every time a publisher then publishes an event with the broker, the broker checks its subscriptions. It sends out notification statements in accordance with indicated preferences to those consumers adopting push based notification; while it retains the events for those consumers notified of events using the pull based mechanism.

This approach has the advantage that a specialized component can be employed to facilitate notification alleviating the need for incorporating support thereof directly into business processes. Moreover, it allows a separation between the events published by a business process and the manner in which interested parties are notified. At the same time from the perspective of the business process it remains possible to give instructions on how an event notification must take place, e.g. that an event must be immediately reacted to informing a broker that notification for this event should be done in a push based manner to at least one subscriber. This illustrates that a broker based notification scheme allows a business process to push its events to the broker without having to consider the actual delivery thereof; while at the same time retain the capability to indicate instructions concerning such delivery.

#### 4.2.1.4. Tool support

When it comes to tool support for monitoring, we find the same ideas as described so far. To mention a few, from the commercial domain tools include the BEA Logic Event server, the stand-alone (or integrated) IBM Tivoli Business Application Manager (<http://www-306.ibm.com/software/tivoli/solutions/business-application-management/>) and IBM's WebSphere Business Monitor V6.1 integrating monitoring functionality into its WebSphere suite (both based on the Common Base Events data model), the TIBCO OpsFactor ([http://www.tibco.com/software/business\\_activity\\_monitoring/opsfactor/](http://www.tibco.com/software/business_activity_monitoring/opsfactor/)) allowing integration of TIBCO's monitoring component with its TIBCO BusinessWorks component for process orchestration, and Software AG's business activity monitoring products part of its Business Infrastructure Suite (<http://www.softwareag.com/Corporate/products/wm/bam/default.asp>).

These commercial tools have in common that are all based on CEP techniques and methodologies and are typically part of larger suites for business process management. Tool development in academia has focused more on event stream processing (ESP), thus emphasizing different kinds of requirements and objectives. Work was done in this context at Stanford, which resulting in its tool STREAM (<http://infolab.stanford.edu/stream/>) and is still being continued at Berkeley in the Telegraph project (<http://telegraph.cs.berkeley.edu/>). Differently, the tools Aurora (<http://www.cs.brown.edu/research/aurora/>) and its successor Borealis (<http://www.cs.brown.edu/research/borealis/public/>) (developed in a collaborative effort between Brandeis University, Brown University, and MIT) take both real time and past events into account. These tools focus on providing implementations of event stream processing algorithms that are capable of real time, continuous monitoring of events from the present world, of archived events, and both in conjunction. A final interesting open source tool is Esper (<http://esper.codehaus.org/>), which is capable of supporting both CEP and ESP.

#### 4.2.2. Payment

The payment compliance concern relates to the definition of payment related requirements, e.g. to comply with MIFID regulations concerning the payment specifics of provided financial services. Also in e-business and e-commerce settings it is important for organizations to be able to express for example whether payment is refutable and annulable for performed activities, whether off line or on line payment techniques are employed, what the settlement model is, and which particular style of pricing is used. Early work in payment comes from the field of payment solution development. An example is [MED93], which proposes NetCash, a framework for real-time electronic payments with provision of anonymity over an insecure network. Although some dimensions of payment are mentioned, such as on line versus off line payment, no types of payment requirement are explicitly identified. Interestingly, the paper does argue for the importance of anonymity (privacy) and security in payment transactions. This implies that the payment concern not only affects the basic business process structure, but also that other compliance concerns can be applicable to the payment concern. This suggests the need for a modelling approach that caters for application of compliance concerns to other compliance concerns.

[NM95] also raises these concerns and adds convertibility of payment means, flexibility of payment specifics, and efficiency to the list of requirements for payment. In addition, [NM95] discusses several approaches for payment including electronic currency systems, credit-debit instruments and secure credit card usage. Although in passing the work implicitly covers payment characteristics (like tangible versus intangible payment means), these are not made explicit. More comprehensively in that regard, [MW97] presents a comparison of (at that

time) recent payment mechanisms. Though the discussed payment mechanisms are relatively old, the criteria used for comparison are of interest here. [MW97] defines the following criteria to evaluate and select a payment mechanism: easily exchangeable, locally scalable, and acceptable to users, low transactions delay, low transactions cost (micro and large transactions), low fixed costs (for seller), non-refutable, transferable, financial risk (for buyer and seller), unobtrusive, anonymous (buyer and seller), immediately re-spendable, privacy and two-way usage. These are not concrete in nature; however they do give a ball park indication of the type of issues to consider for payment requirement modelling. Similarly, [PAS+98] explores types of payment model to establish the requirements for a generic payment service. Identified models include account-based and token-based payment models. Also, the need for security is emphasized.

Surprisingly, work on payment in the context of service oriented computing has been very limited. [SEH02] introduced several price and payment models, but its focus is limited to the specification of low level characteristics such as the specific payment instrument to be used. Recently, more extensive work has been reported in [OY08], which presents a multi-level classification of payment requirements. The classification identifies payment criteria, mechanisms and measures; and moreover makes explicit the dependencies between these criteria, mechanisms and measures. Criteria that are considered are: value, negotiable, payment means (financial, natural), settlement model (e.g. escrow or metered), annulable, refutable, immediacy, re-spendable, transferable and traceability. These are themselves linked to specific payment mechanisms. For example, the value is refined in a price while the negotiability of the payment amount influences the pricing style (e.g. absolute or ranged). Thus, choices for a particular payment criteria influence the possible mechanisms to be adopted. In a similar fashion the choice of payment mechanism affects the specific payment measures. For example, if payment must be immediately received, then this places restrictions on the type of payment instrument (as traveller cheques for example take time to process).

### 4.2.3. Privacy

The privacy compliance concern encompasses compliance requirements related to privacy in business processes regarding information and/or resources. In some cases, such as companies conducting surveys, the anonymity of participants must be guaranteed while participating in activities. For example, customers can demand to have information in order to determine the trustworthiness of an organizations like concerning its reputation, reviews by others, and so on; and related stipulate what level is trust is acceptable and which is not. Similarly, organizations may require particular information from their customers to establish that their incoming orders can be trusted (e.g. based on provision of credit information). Customers might agree on doing so, but only under the assurance that such information is not disclosed to others. A significant body of work has been performed on the analysis, specification and enforcement of privacy in relation to web services.

One group of works can be found in the semantic web corner. For example, [TDT05] describes a semantic-based user privacy protection framework for web services. Their framework is an adaptation of the work done by the Platform for Privacy Preferences (P3P, <http://www.w3.org/TR/P3P/>) with regard to the purpose for which user data is collected, the recipient of this data and the retention thereof. Concretely, in the framework web services define their input parameters and whether they are required or not. Users at the same time define how much of their personal information they wish to be made available to the services. Encoding of the user privacy preferences is done in terms of different permissions levels grounded on a domain specific service ontology based on DAML-S. Three levels of data

permissions are identified, being free, limited (for execution purpose only), and not given. Based on these preliminaries [TDT05] then describes how negotiation can be facilitated to identify problems, like data required by a service but not given by an user; essentially trying to determine the data elements that can be exchanged between the parties, without violating the users privacy preferences. The paper also mentions the process of obfuscation to meet privacy requirements by making data less specific (through abstraction and/or falsification of information).

Similarly, [KS02] also identifies purpose, disclosure to third parties and retention as relevant issues. They model these in terms of obligations to establish a privacy policy model for enterprises. Continuing, [ROB+02] provides a framework for preserving privacy in web services. A difference is made between user privacy, service privacy and data privacy. User privacy falls into sensitivity of information, who receives the information and for what purpose(s) it is used. Service privacy also deals with usage, but in addition covers storage of information (particularly how long information is kept), and if and how information is disclosed to third parties. These elements also come back in the work at (<http://www.w3.org/TR/P3P-preferences/>), which defines a P3P Preference Exchange Language (APPEL). This language was originally aimed at enabling web sites to describe their privacy policies, but has recently been in an adaptation process to also relate to web services. It identifies several issues of importance for privacy. Besides the ones already mentioned, it observes the importance of stating how disputes can be reported, and how these are then remedied.

Interestingly, in the mentioned works a re-occurring observation is that privacy requirements and retention of information are intertwined. This can potentially lead to conflicts; e.g. that information must be retained to be compliant which an user refuses to give. This in turn implies a mechanism for prioritizing compliance requirements. We return to this matter in section 5. Having said that, works like [YKS06] have taken privacy regulations (such as HIPAA, the European Privacy Directive and Canadian Privacy Legislation) as their starting point. These works have in common that based on the legislations they extrapolate an underlying set of privacy issues. [YKS06] identifies five issues: 1) who wishes to collect the information; 2) what is the nature of the information; 3) what is the purpose which the information is being collected; 4) how long is the information kept; and 5) to whom can the information potentially be disclosed. It then proposes a privacy model to capture these in a minimal set of constructs, which can then be mapped to e.g. APPEL.

With another emphasis, [KFP+04] discusses the issue of privacy in realization to its enforcement through employment of security related technologies. This is relevant as it implies a co-dependency between security compliance concerns on the one hand and privacy compliance concerns on the other; where privacy requirements depend on enforcement of security requirements for their realization. This link is also established in the EPAL language, which explores the usage of access control mechanisms to achieve data privacy protection. The proposed language, short for Enterprise Privacy Authorization Language (<http://www.zurich.ibm.com/security/enterprise-privacy/epal>), relies heavily on XACML ([http://docs.oasis-open.org/xacml/2.0/access\\_control-xacml-2.0-core-spec-os.pdf](http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf)) which is a standard for security policy specification (discussed in more detail shortly in section 4.6). Although EPAL seems to have lost momentum, work in this regard has been continued in the form of a privacy profile for XACML based policies. In the same spirit [CFH06] explores possibilities for privacy enforcement through access control, cryptographic encryption and hash based protection of information. It also gives an extensive overview of related work in this area. Implementation oriented, [JG07] describes an IBM tool based approach to implement a privacy auditing framework with a so-called Hippocratic Database. Essentially

the approach focuses on data protection through appropriate usage of authorization and confidentiality techniques; where moreover privacy related events can be monitored and audited.

#### 4.2.4. Quality

The quality compliance concern covers preferences and demands concerning the level of quality in business processes. Quality in this regard is a broad concept and is related to the quality level maintained within a process. Examples of this include customer preferences regarding the speed with which their requests are handled, the accuracy of financial reporting data, the level of availability for mission critical services during an emergency response and so on. Considerable work has been done on the development of so-called quality description languages for services, as the default description language for services, WSDL, provides no support for quality specification. Web Service Level Agreement (WSLA) defines a standard for defining service level agreements. It identifies several quality related parameters; however these are low level in nature. [SEH02] mentions several quality related requirements for services, but its focus is mostly on the usage of benchmarks and standards to express such requirements. Similarly, [CSM+04] and [ZBN+04] also only consider a limited set of quality dimensions in relation to composition of QoS (Quality of Service) enabled services.

Work by the standardization body OASIS identifies several quality related requirements. The so-called Web Services Quality Model (<http://www.oasis-open.org/committees/download.php/15910/WSQM-ver-2.0.doc>) allows service providers and requesters to extend their WSDL definitions with quality properties. They define a classification of quality properties grouped along the business, service and system level. The earlier mentioned Service Component Architecture (SCA) Policy framework specifies a policy specification framework for service components in which so-called intents are used to express abstract non-functional requirements. With regard to quality, the intent for reliability is identified and several concrete items such as ordering of message and delivery semantics are defined. More comprehensively, [FK98] describes a specification language, QoS Modeling Language (QML), for specifying QoS. QML is an extension of the Unified Modeling Language (UML), and includes concepts of QoS contract types and contracts. QML is a generic language considering so-called contract types such as performance and availability. These types are then made concrete in dimensions, e.g. throughput for performance. The approach also allows verification of horizontal consistency between service providers and requesters. However, these works only discuss performance and availability, whereas other types of quality requirements surely exist (such as accessibility). Moreover, they do not make the high level quality requirements explicit, as such not allowing organizations to assess consistency of quality requirements at different levels of abstraction.

In the field of automated negotiation [CP05] describes an approach using a negotiation broker to which both the consumer and the service provider can notify their preferences on QoS attributes and negotiation strategies by specifying the value of a relatively small set of parameters. However, the amount of properties considered is limited and not on multiple levels of abstraction. Related, [YL04] suggests a QoS-capable Web service architecture by introducing a QoS broker module between service clients and providers. The focus is on developing decision and negotiation algorithms in response to service requests. Unfortunately higher level quality requirements are also not considered in these algorithms. As such, the extent in which quality modelling can be supported is limited.

A possible solution to addressing this limitation is described in [OY08b], which proposes a multi-level approach to quality specification. It identifies six categories of quality

requirement: accessibility, availability, accuracy, reliability, and performance (encompassing efficiency and responsiveness). Constructs to capture these different types of requirement at multiple abstraction levels are then introduced; linking concrete quality measures to high level objectives and vice versa. These links can be made explicit by organizations using so-called consistency rules; which in turn allows one to automatically assess whether objectives are consistent with low level measures. This capability is of importance for compliance verification.

#### 4.2.5. Retention

An important part of organizational compliance revolves around retention; that is, the proper archiving, retrieval and disposal of information. As section 3 demonstrated legislative regulations such as Basel II, FINRA-SEC, HIPAA and Sarbanes-Oxley impose constraints on the retention of information by organizations. For this reason organizations are looking for ways to reduce the risk associated with managing growing and disparate forms of data, while at the same time meeting the requirements for regulatory compliance, corporate governance, and litigation support. All of these can result in large fines and/or business losses if they are not met. For example, during litigation organizations must be able to quickly accurately produce requested information. If this takes too long or information can not be produced at all, then organizations face hefty fines. Moreover, if information has been intentionally destroyed, then criminal charges may also be brought against those held accountable by law. In order to avoid these dangers organizations typically employ a wide range of both organizational and technological activities. Standardization efforts have been made in this regard in the field of records management. Records are defined in this context by the International Standards Organization (ISO, <http://www.iso.org/iso/home.htm>) as "information created, received, and maintained as evidence and information by an organization or person, in pursuance of legal obligations or in the transaction of business".

Several ISO adopted and promoted standards have been developed in the area of records management. These include ISO 23081 ([http://www.iso.org/iso/catalogue\\_detail?csnumber=40832](http://www.iso.org/iso/catalogue_detail?csnumber=40832)) which deals with records management processes and a meta data model for records, ISO 15489 ([http://www.iso.org/iso/catalogue\\_detail?csnumber=31908](http://www.iso.org/iso/catalogue_detail?csnumber=31908)) which concerns records management, and ISO 15836 ([http://www.iso.org/iso/iso%20catalogue/catalogue\\_tc/catalogue\\_detail.htm?csnumber=37629](http://www.iso.org/iso/iso%20catalogue/catalogue_tc/catalogue_detail.htm?csnumber=37629)) which comprises the so-called Dublin Core meta data element set. ISO 23081 identifies six types of meta data: 1) meta data about the record itself such as an identifier and time stamp; 2) meta data about the business rules/policies that mandate and govern the retention of the record; 3) meta data about the agents involved in the retention and receipt of the record; 4) meta data about the business activities or processes responsible for creating and/or modifying the record; 5) meta data about the records management processes like followed procedures, time of initiation, and so on; and 6) meta data about the meta data record for example the format used. Differently, ISO 15836 defines a set of data elements aimed at facilitating the discovery and subsequent retrieval of records (also via indexing of records based on content and/or meta data). Examples include key words, category statements using well defined taxonomies, etc, to enable proper record description. Finally, ISO 15489 gives a more general introduction to records management

The scope of these ISO standards covers the records management environment as a whole including both paper based and electronic records. As the focus of this report is on computerized, service enabled business processes, available techniques, methodologies and

tools for electronic records management are of particular interest. Solutions that have become increasingly popular for addressing electronic information management issues constitute so-called electronic document management systems. An electronic document management system is a computer based system that provides the capabilities necessary to track and store electronic documents; where the potential range of targeted electronic documents varies from emails to business documents to system events. [SPR95] defines electronic document management systems as "the creation, storage, organization, transmission, retrieval, manipulation, update, and eventual disposition of documents to fulfil an organizational purpose". This definition covers three basic issues: 1) storage of information, 2) access to information, and 3) destruction of information. All are equally important in order for organizations to ensure that the information flowing within and among their business processes is handled in compliance with legislative regulations.

Retention of information deals with the actual storage of information. This typically can comprise a wide diversity of information types, such as emails, documents, images, events, videos, and etceteras. Moreover, retention not only concerns the storage of the actual information, but especially also meta-data about this information. This meta-data can include a wide variety of data. This matter is explored within the Moreq2 (Model Requirements for the management of electronic records) framework ([http://www.cornwell.co.uk/moreq2/MoReq2\\_body\\_v1\\_02.pdf](http://www.cornwell.co.uk/moreq2/MoReq2_body_v1_02.pdf), [http://www.cornwell.co.uk/moreq2/MoReq2\\_appendix\\_9\\_v1.02.pdf](http://www.cornwell.co.uk/moreq2/MoReq2_appendix_9_v1.02.pdf)) commissioned by the European Union. These specifications describe requirements for electronic document management systems in order to establish a European wide benchmark against which electronic document management solutions can be compared. Based partially on ISO 23081, Moreq2 identifies meta-data parameters for documents in several categories such as identification, description (including relations to other documents), categorization, location and retrieval attributes, history tracking and versioning characteristics, record types, tamper proof mechanisms, and retention and disposition related data. Similarly, the UK National Archives (<http://www.nationalarchives.gov.uk/documents/metadafinal.pdf>) has developed a meta-data standard that defines a set of characteristics. In addition to Moreq the standard also identifies issues such as the format, language, syntax and semantics as important when retaining information, as well as the preservation of documents (to ensure future readability). Items akin to the ones mentioned can also be found in the US variant of the requirement set for electronic document management systems (<http://jitc.fhu.disa.mil/recmgt/p50152stdapr07.pdf>), developed by the US Department of Defence.

In relation to compliance and business process modelling the above discussion is important to the extent that retention, retrieval and disposition of the information flowing within business processes must somehow be supported via the employment of electronic document management solutions. Relevant work has been done already in this regard on the integration of document management capabilities directly into other applications (e.g. using LDAP or SOAP communication). This has the advantage that users may retrieve existing documents directly from the document management system repository, make changes, and save the changed document back to the repository as a new version, all without leaving the application. This seems to be a natural approach to take for business processes as well, since it allows processes to remain agnostic about the actual activities performed to facilitate the storage, retrieval and destruction of information. The only requirement is that it must be clear what kind of parameters business processes have to provide in addition to the actual to-be-stored-information in order for this to take place in a correct manner. Also, another important aspect of this is the specification of data retention, access and destruction policies. Such policies

capture (among others) under what circumstances certain information is to be captured, for what period, at which location, and destroyed in what manner. This relates to what ISO 23081 refers to as the business rules and policies mandating and governing the retention of information. In relation to compliance and business processes these rules and policies find their origin in the legislative regulations discussed in section 3, and should be explicitly captured.

#### 4.2.6. Security

The security compliance concern pertains to a wide range of issues in order to adequately protect processes and the resources used by these processes. For example, in the health care setting a key issue is that private patient information is not shared with insurance companies; that is, that information is not disclosed to unauthorized parties. Another example is a requirement concerning the usage of specific encryption protocols to securely communicate financial auditing data. Current work in service oriented computing and web services related to security mostly focuses on developing technologies that provide message level security in order to protect individual electronic messages. Standards such as XML Signature (<http://www.w3.org/TR/xmlsig-core/>) and XML Encryption (<http://www.w3.org/TR/xmlenc-core/>) provide low-level support for signing XML messages. XML Signatures provides integrity, message authentication and/or signer authentication services for data of any type. XML Encryption is a standard for a process for encrypting/decrypting digital content (including XML documents and portions thereof) and the XML syntax used to represent it. Both standards operate directly on XML documents, and are schema independent. Web Services security standards, such as WS-Security, WS-SecurityPolicy, WS-SecureConversation, etc. are relatively at higher level in nature. WS-Security, which relies on XML Signature and XML Encryption, proposes a standard set of SOAP extensions that can be used when building security Web Services to implement integrity and confidentiality, supporting multiple tokens, trust domains, signature formats, etc. WS-SecurityPolicy provides the mapping of the WS-Security standards onto the infrastructure proposed by the WS-Policy standard. WS-SecureConversation is built on top of the WS-Security and WS-Policy models to provide secure communication between services. WS-Security focuses on the message authentication model.

All of the standards which have been discussed have common features in the sense that they focus heavily on developing technologies for establishing privacy and integrity of electronic messages. As such, they fail to provide a comprehensive, conceptual approach for the specification of technologically related security requirements, neglecting the need for authentication and authorization. An exception is [LPH04] which presents an event based security framework for authentication and authorization. However, like the other works it does not address the definition of higher level, business oriented security motivations and requirements into consideration. WS-SecurityPolicy allows people to express assertions concerning security, however the properties that can be specified are directly related to other specifications (like WS-SecureConversation) and as such they are low level and technology dependent in nature. [SEH02] identifies several conceptual security related properties, but does not consider higher level security requirements. The Service Component Architecture (SCA) Policy framework (<http://docs.oasis-open.org/opensca/sca-policy/sca-policy-1.1.html>) specifies a policy specification framework for service components in which intents are used to express abstract non-functional requirements. With regard to security, intents for authorization, confidentiality and integrity are considered. However, how these are exactly mapped to concrete types of technology is not addressed. The WSQM, discussed earlier in relation to quality, relates security mechanisms with particular technologies, but it fails to

provide support for formal verification of dependencies. It also does not make the reasons for security, i.e. prevention of security threats, explicit.

The eXtensible Access Control Markup Language (XACML, <http://docs.oasis-open.org/xacml/2.0/access/control-xacml-2.0-core-spec-os.pdf>) describes a sophisticated language for security policy specification. It provides a core schema for the expression of authorization policies in XML. It relies on the Security Assertion Markup Language (<http://docs.oasis-open.org/security/saml/v2.0/>) to define the assertions to be used within XACML policies. Although these works are certainly relevant, they focus on authorization only. They also do not consider security requirements at different levels of abstraction. A work that does consider this matter is [MG03], which defines an ontology for modelling security in relation to the Tropos approach. The work considers security from the early phases of requirements analysis to the stages of system development. Concretely, several security-related constructs are introduced to express security constraints. The approach allows organizations to express so-called secure goals, which can then be realized via secure tasks. The work is interesting in nature, yet only provides some examples of security goals and tasks.

#### 4.2.7. Transaction

Finally, the transaction compliance concern involves transactional requirements for business processes. [PAP03] presents an extensive overview in this regard. It perceives a business process as a composition of business transactions. Each of these transactions represents a consistent change in the state of the business. The paper identifies two types of transaction: atomic transactions and long-running transactions. Atomic transactions conform to the classical ACID properties originating from the database transaction field; and can be nested in a close nesting model (performed all together or not at all). In contrast, long running business transactions are more relaxed in nature. They essentially constitute groups of atomic related transactions in an open-nested style. In such open nested style transaction each of the atomic transactions represents a milestone signalling partial completion of the overall transaction. Participants can decide independently whether or not to commit results of these transactions, and as such the outcome of the transaction need not be the same for all participants.

Since results are committed without knowing the final result of the overall transaction, these results are no longer isolated. Therefore, to ensure that the business process remains in a consistent state some form of recovery is needed. Backward recovery is one option, and entails that the process performs activities to undo all the work done. For atomic transactions this can be done in an automatic manner, since no results are made durable before a commit has occurred. In contrast, in long running transactions partial results may have been stored (and used by other parts of the process/other processes). [TIR+03] makes the valid point in that regard that in the context of service-oriented computing based business collaboration, backward recovery would imply locking of resources as well as presence of pre-defined compensation operations for each operation of a service. Both assumptions are often not realistic. Therefore long running transactions alternatively can use forward recovery in which compensating activities are carried out to undo the effects of already performed work (rather than undoing the work itself).

An initial option that has been explored in service oriented computing, was to build transactional capabilities directly into the business process language. Most notably, WS-BPEL provides compensation mechanisms that are grounded on the notion of scopes (activity nesting). Although this offers some support, [CSM07] notes that the disadvantage is that in such event business process logic and transaction logic are mixed. Appropriate compensation

handlers must be defined for example as well as scopes, which implies that the process developer has intimate knowledge of all the transactional implications. Moreover, WS-BPEL lacks support for isolation of results. Also, the WS-BPEL process is responsible for both initiating and coordinating the transaction. Furthermore, coordination is done locally at the WS-BPEL process and as such failures may occur at a partner of which the process is unaware. This can lead to an inconsistent state of the process. To remedy this situation modifications to WS-BPEL would be required that affect its semantics.

For this reason there has been extensive work on providing the above described transactional models. WS-Transaction (<http://dev2dev.bea.com/pub/a/2004/01/ws-transaction.html>) defines protocols for transaction processing on top of the WS-Coordination (<http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.1-spec-errata-os.pdf>) specification. In a WS-Coordination protocol there is a coordinator and a set of participants involved in a distributed activity. Participants register with the coordinator through a registering service. During execution the coordinator passes a so-called coordination context to the participants involved to keep them informed about the progress. WS-Transaction applies these concepts to the area of transactions. On the basis of WS-Transaction particular transaction protocols have been defined for atomic transactions and business activities (long running transactions).

Atomic transactions are described in Web Services Atomic Transaction (<http://docs.oasis-open.org/wstx/wstx-wsat-1.1-spec-errata-os.pdf>), defining a two-phase commit and completion protocol. In a two-phase (2PC) commit the coordinator ensures that registered participants reach a commit or abort decision for a transaction, and ensures that all participants are informed of the final result. This 2PC protocol has two variants: volatile 2PC in which participants are managing volatile resources such as a cache register; and durable 2PC in which participants managing durable resources such as a database register. The completion transaction builds on these two variants to first obtain an initial commit for a transaction from all participants, and subsequently notify all participants to finalize the commitment.

Web Services Business Activity (<http://docs.oasis-open.org/ws-tx/wstxwsba-1.1-spec-errata-os.pdf>) provides two protocols for long running transactions (also grounded on WS-Coordination). The first one is called "Business Agreement participant relies on the coordinator to inform the participant when it has received all requests to perform work within the business activity. Thus, in this protocol the coordinator indicates when the transaction is done. With Participant Completion". In this protocol a participant registers with the coordinator, so that the coordinator can manage it. A participant knows when it has completed all work for a business activity, and informs the coordinator when this is the case. In the "Business Agreement With Coordinator Completion" protocol a participant also registers with the coordinator. However, the participant relies on the coordinator to inform the participant when it has received all requests to perform work within the business activity. Thus, in this protocol the coordinator indicates when the transaction is done. The outcome of a business activity can be atomic in nature (closed nested transaction) or have a mixed outcome (in which case some participants may commit results while others have to undo/compensate activities).

Usage of the above solutions typically concerns layering WS-Transaction protocols on top of WS-BPEL processes. For example, [TKM04] suggests usage of WS-Policy assertions to declaratively attach transactional coordination constructs to WS-BPEL partner links and WS-BPEL scopes. These constructs are based on those defined in Web Services Atomic Transaction and Web Services Business Activity and indicate particular transaction protocols to be followed. A partner link with an attached coordination policy is called a coordinated

partner link. Such a link defines the abstract requirement on any (concrete) deployed Web service that aims to provide the partner functionality during execution. By attaching similar information to WSDL descriptions it can then be ensured that only partner services are selected that support the stipulated transaction model(s). The WS-BPEL scopes (i.e. groups of activities) are also extended with appropriate constructs to declare transactional semantics. During execution a generic transaction service is then engaged to enforce these semantics.

Somewhat similar, [CSM07] describes an aspect-oriented based approach to integrate a generic transaction service into WS-BPEL. This transaction service then handles the coordination and ensures that the coordination context is included in the exchanged messages. This allows the integration of a wide variety of transaction models into BPEL without affecting its semantics; where the only limitation is the level of sophistication of the used transaction service. [CSM07] uses a web service based on the open source Apache-Kandula that implements both WS-AT and WS-BA. Additionally, [CSM07] suggests using a persistence service to save original values in case atomic transactions must be undone. The difference with [TKM04] is that in [CSM07] attachment of transactional requirements is done in externalized scripts rather than inside the WS-BPEL process definition. This allows for a more manageable solution to both process definition and transactional requirements. This is particularly of interest as transactional requirements typically only constitute a small part of the compliance requirements applicable to business processes.

A matter not considered in these works is how transaction models relate to other compliance concerns. This issue is raised in [PAP03], classifying atomicity in non-repudiation, conversation, payment, goods and certified delivery. Non-repudiation deals with digitally signing the content of a transaction at an application level. This constitutes a combination of transactional and security requirements. Conversation atomicity is more similar to what is described in WS-AT and WS-BA, and deals with the atomicity of partner interactions. Contract atomicity extends this by adding a legally binding interpretation to the transaction (which potentially can be another compliance concern to take into consideration). Payment atomicity builds on this to ensure that payment transpires in a correct manner thus combining payment and transactional requirements.

Continuing, goods atomicity is payment-atomic, but also implies that the goods will be received only if payment has been made. This constitutes a combination of payment, transactional and functional requirements. Finally, certified delivery atomicity constitutes a goods-atomic transaction with the additional requirement that the right goods are delivered (essentially adding a confirmation activity to this regard, i.e. another functional requirement). From a compliance point of view these combinations are interesting, as it allows compliance goals to be expressed in abstract terms that can then be made more concrete into several types of requirement. [MK06] proposes an initial framework for this in which these higher level transaction requirements are mapped onto (combinations of) lower level, basic transactions models. It would be interesting to see if this can be combined with [CSM07] for example to construct a transaction service, which supports high level specification of transaction requirements internally supported by appropriate low level transaction protocols.

### 4.3. Conclusions

In the previous two sections an analysis of current modelling solutions for capturing both basic and advanced compliance concerns was presented. The overall message of this analysis is that in order for business processes to be compliant, the specification of an extensive

number of different types of requirements must be catered for. It is not difficult to imagine that as business process descriptions incorporate more and more compliance requirements, they become more and more complex in nature. This necessitates adoption of a modelling approach that is modular in nature making it possible to easily manage business process descriptions. “Within the COMPAS project such approach will be realized in the form of the Model-driven Integration Architecture for Compliance. In this architecture business processes will be describable via views that are refinement of generic core views (defining constructs such as element, attribute and reference). Each of these views captures a particular aspect of a business process, displaying only those parts of the process relevant for the view (while ignoring all others. This includes basic views such as the process control flow view, but also more advanced views such as the retention view (encompassing e.g. retention activities, repository locations, storage records, and etceteras). New views will be easily addable, which is useful as compliance requirements hitherto not considered in this deliverable may need to be addressed in the future.

The contribution in this regard of the discussion in section 4 lies in the fact that it examines the current literature to find out what constructs are available to describe both basic and advanced compliance concerns. Moreover, it does so for these concerns (where possible) from both a business and technological perspective. These can potentially form a point of departure for the development of views for the identified compliance concerns using the methodologies, techniques and tools provided by the Model-driven Integration Architecture for Compliance. Particularly, it is likely that each concern leads to description within multiple views. For example, the control flow concern can be expected to be described in a business view (e.g. in terms of activities) as well as a technological view (e.g. expressed in terms of service operations). Similarly, the retention concern will be captured in multiple views as well providing constructs to express both in business and technological terms what information should be retained, how, by whom, where, and when. These constructs themselves can be defined within one or more domain specific languages (DSLs).

An interesting observation in this regard is that these DSLs must be such that they allow compliance requirements to be not only specifiable in relation to the basic compliance concerns, but also in relationship to each other. For example, the DSLs should allow expression of statements such as that an event must be logged in a secure manner. This not only involves defining a DSL for the retention concern that builds on the DSL for the information concern, but also that constructs in the DSL for the retention concern can be combined with those within the DSL for the security concern to express that logging must be done such that any tampering is detected (as such preserving the authenticity of logged events).

Finally, the literature review thus far has only partially addressed compliance specification. As observed in section 2.6 compliance specification not only requires the need for process models that can express compliance requirements, but also necessitates the existence of policies and rules to explicitly describe the compliance requirements to which business processes must conform (and under which circumstances). In the following section, section 5, relevant work in this area of compliance specification is examined. Section 5 also analyses potentially interesting solutions for 'compliance improvement' enhancing measures.

## 5. Compliance specification and improvement

In the previous section an analysis of current modelling solutions for capturing compliance concerns was provided of relevance for the different compliance legislations discussed in section 3. In this analysis the focus was restricted to the modelling of compliance requirements within business process models. As observed though in section 2.6, compliance specification also entails capturing the compliance policies and rules applicable to business processes. In this section a large body of work is reviewed specifically aimed at or potentially suitable for providing the means with which compliance policies and rules can be specified. Throughout this review relevant functionalities for a policy/rule language offered by individual solutions are extrapolated. Specifically, approaches from several domains are analysed; being rule based approaches, policy based approaches and goal based approaches in subsections 5.1 through 5.3 respectively. As a final remark please observe that the following review is not intended to be exhaustive in nature, but rather is aimed at discussing influential and relevant works for compliance requirement specification. In the different subsections potentially useful notions within these works are marked by the **text** representation.

### 5.2. Rule based approaches

In rule based approaches the idea is that compliance requirements are defined as rules expressing requirements, while the to-be-verified business process is viewed as a collection of facts. Then, in order for the business process to be compliant these approaches in one way or another assess whether its model satisfies the goals. This idea has for example been explored in [LMX07] and [SGN07]. In such works verification can then either be done in a reactive manner to identify occurred violations, e.g. by employing methods from the area of model checking (like SPIN [HOL03]), process algebras (like REO [ARB04]) and theorem proving (e.g. [RKM06]). This is useful for auditing purposes, however ignores the fact that in today's business environment organizations need to be able to control their business processes such that violations are avoided; and in the event that violations do occur, an immediate response can be carried out.

Pro-active enforcement of compliance requirements is as such an essential complement to after-the-fact verification where compliance controls are integrated into business processes while they are executing. Although not much work has been done yet in this regard in compliance, in other fields interesting approaches have been developed. For example, [ZBL+03] describes a rule based inference framework DYflow in which rules are used to drive the development of service compositions. Depending on the specific requirements services are composed on the fly, where decisions concerning their ordering and binding are governed by rules. In the following the current literature on rules is analysed, where for structure purpose this analysis is split into solutions for rule specification, analysis, and implementation.

#### 5.2.1. Specification

Rule languages are languages that allow expression of compliance requirements in the form of rules. Rules are accepted principles or instructions that states the way things are or should be done, and tells you what you are allowed or are not allowed to do. Loosely speaking, rules are typically thought of and expressed as if-then statements. An example of a typical compliance rule is that "if a sales employee is involved in processing an order, then he/she can not also approve the order". Another is that "if authentication for a core business

operation fails, then an appropriate event must be logged and the security manager is notified". A first group of rule specification languages we consider stems specifically from the area of web services and service oriented computing. Employed increasingly nowadays for the realization of inter-organizational business processes, these languages focus on the description of the capabilities of services offered and consumed by different parties. These languages include Web Service Level Agreement (WSLA) [DDK+04], Web Service Modeling Language (WSML, <http://www.w3.org/Submission/WSML/>) and Web Service Offerings Language (WSOL) [TPP02]. WSLA was designed to capture service level agreements between parties in a formal way to enable automatic configuration of both the service implementing system of the service providing organization as well as the system that is used to supervise the agreed quality of service. WSOL takes a service provider point-of-view focusing on the formal representation of a web service in terms of various constraints (such as pre-, post-, and future -conditions, quality of service constraints, access rights, price/penalty and management responsibility).

Both languages are **symmetric** in nature, which allows organizations to depict to what requirements they can comply in the same manner as compliance sources can define their expectations (for example to enable service providers and requesters to reason about and negotiate quality of service policies). They are also XML-based languages allowing them to be **easily communicatable** and (potentially) **interoperable**. This makes the exchange of compliance requirements easier. It also makes these requirements **parseable** and thus understandable for computerized systems. WSOL and WSLA furthermore provide **conceptually naturally semantics** as their rules are directly grounded on web service based concepts. This makes it more intuitive for developers in an organization what compliance requirements mean, so that they can specify them in a more convenient fashion. It also enables them to potentially address the different compliance concerns identified in section 3 (assuming that the web service concepts are enriched with constructs as discussed in section 4). WSOL furthermore allows grouping of its constraints into so-called constraint groups. This kind of **structuring** mechanism is useful, as the number of requirements to which organizations must comply, is typically very large. As such, the option of grouping together logically related compliance requirements will enhance their manageability. In relation to compliance these groups can be used to describe a certain course of action by defining a set of logically related requirements such that they constrain (some part of) the business in a coherent, consistent and meaningful manner. To exemplify, in the first pillar of Basel II several methods for credit risk assessment are proposed leading to different computations (and thus different sets of compliance requirements).

WSOL furthermore allows constraint groups to extend one another via single inheritance to facilitate **reuse** in order to further enhance manageability. WSLA offers a similar feature for grouping using so-called obligation groups. These obligation groups also reflect that WSLA uses the notion of obligations to give extra weight to requirements. Such notion of **modality** contributes to a requirement's expressive power, allowing one to use modal expressions to qualify the truth of a judgement. There exist different kinds of modal logic including alethic logic (necessity and possibility), deontic logic (obligation, permission and forbidden) (<http://plato.stanford.edu/archives/win2004/entries/mally-deontic/>), temporal logic (e.g. it has always been the case and it will eventually be the case) (<http://plato.stanford.edu/archives/win2003/entries/logictemporal/>), and doxastic logic (belief and non-belief). Such statements are useful for organizations to give extra weight and nuance to their compliance requirements. For example, there is a difference between the requirement that it is necessary that an auditing task is performed and that it is permissible to do so (conveying a different sense of urgency). Like WSOL, obligation groups can contain other

groups to facilitate reuse. Finally, the expressiveness of the rules themselves in WSOL and WSLA is equivalent to standard predicate logic while offering several **operators** such as numerical comparison.

In contrast, WSML is a full-fledged rule specification language developed for the Web Service Modeling Ontology (WSMO, <http://www.w3.org/Submission/WSMO/>), offering different variants catering for different complexity needs making the language **customizable**. This is useful as not all compliance requirements will require equal expressive power. Its full variant supports several advanced features such as quantification, negation, monotonicity and the specification of unsafe rules. Quantification is typically thought of in terms of the first order logic operators of 'for all' and 'exists', and enables organizations to define compliance requirements like 'all financial processing tasks must be followed by a control task' and 'there must exist an external auditor that is involved in the financial reporting process'. The possibility to specify **negated** statements is useful for compliance requirement specification as well as it often concerns things an organization should not do. For example, it must not be the case that an employee has custody of services and also accounts for them; since in such situation there is a high risk of that person using the services for personal gain and adjusting their logs to cover theft. Also, following privacy legislation it might be that it must not be the case that information is retained.

A useful distinction in this regard is the intent of the negation, which can be strong or weak in nature. Strong, or classical, negation conveys the necessity to explicitly show that something is not true. In contrast, negation interpreted in a weak sense indicates that something is considered not true if it cannot shown to be true. This is a subtle yet important difference for compliance, as in one case explicit proof is required of separation of duties whereas in the other case this is not necessary. In technical terms weak negation assumes a closed world, where knowledge about the world is assumed to be complete; while strong negation takes an open-world assumption in which knowledge is never complete and thus a negative fact must be explicitly proven. WSML supports closed world assumptions only. WSML also considers **monotonicity** of its rules, that is, whether they are monotonic or non-monotonic. Non-monotonic rules (also known as defeasible rules) are common in business, for example to override standard rules with special-case exceptions, to incorporate more recent updates and etceteras (an argument also made in e.g. [ABW04]). In relation to compliance the matter of non-monotonicity is of interest as it allows organizations to indicate to what extent it is important that a compliance requirement is met; and consequently how grave the consequences are in case the requirement is violated. Monotonic compliance requirements must always be met, but non-monotonic ones may be violated (albeit potentially at a cost).

In the realm of generic XML based proposals we find languages like Semantic Web Rule Language (SWRL, <http://www.w3.org/Submission/SWRL/>), Rule Markup Language (RuleML, <http://www.ruleml.org>) and Web Rule Language (WRL, <http://www.w3.org/Submission/WRL/>). SWRL is a proposal for a Semantic Web rules-language, combining sub-languages of the OWL Web Ontology Language (<http://www.w3.org/TR/owl-features/>) with those of RuleML. This gives SWRL the capability to provide for **conceptually natural semantics** as the meaning of the concepts utilised in the specification of rules is defined in OWL. SWRL itself does not support quantification, but its extended version SWRL-FOL (SWRL First order Logic) does. Also, modifiability and clustering were not considered in SWRL's design. Designed to be the new de facto standard for rule specification, RuleML constitutes more an interoperability framework between different (mainly industry) rule languages rather than a language itself. As such, RuleML comprises a collection of languages that serve different purposes with different expressive power. RuleML allows for example for using both classical negation (to

enable open world assumption) and negation-as-failure (closed world assumption). It also caters for specifying unsafe rules depicting which statements are to be closed and which are not. Lastly, first attempts have been made concerning integration of modal logics into RuleML [BOL06]. Finally, the Web Rule Language (WRL) is modular in nature in the sense that it offers different variants with increasing expressive power. As such, like WSML it is **customizable**. Its least expressive variant WRL-Flight essentially is equivalent to Datalog with (stratified) negation using perfect model semantics. In contrast, WRL-Full allows not only for **quantification** and **negation**, but also **non-monotonicity**; where semantics are provided using well-founded semantics [GRS91]. Moreover, unsafe rules may be specified. Like SWRL and RuleML though, WRL provides no support for modifiability other than definition of semantics in ontologies. Also, the issue of how to cluster and manage logically related sets of rules is not explicitly considered.

From the modelling world the Object Management Group (OMG) has developed Object Constraint Language (OCL, <http://www.omg.org/docs/ptc/03-10-14.pdf>) and its proposal for Semantics of Business Vocabulary and Rules (SBVR) language (previously known as Business Semantics of Business Rules, <http://www.omg.org/docs/dtc/06-03-02.pdf>). OCL is a declarative language for describing rules that apply to object-oriented models. It allows for specification of constraints, but not other types of rules such as derivation rules. Constraints can also not be non-monotonic or employ modalities. OCL does allow for both strongly negated and weakly negated constraints. Clustering of constraints into policies is not supported however. A more elaborate language for rule specification from the OMG is SBVR. The aim of this language is to help business people define business vocabularies for business rules of all kinds of business activities of all kinds of organizations. To this end it defines the concepts and semantics of these concepts needed to express business rules. The language has considerable expressive power only limited in its support for modalities (facilitating alethic and deontic logic). Notably, it also allows for explicit specification of closure of facts, that is, whether they abide to closed world or open world semantics (and thus whether facts of compliance that something is not the case must be explicitly proven or not).

Relevant work is also the Rule Based Service Level Agreement language (RBSLA) [PAS05]. RBSLA focuses on knowledge representation concepts for service level management of IT services. At its core are rule-based languages to describe contracts and service level agreements in a formal way, which provides all the mentioned expressiveness features of WSML (excepting the definition of unsafe rules). One of its key features not yet observed is the support for **prioritization** of rules allowing for automatic resolution of rule conflicts. Prioritization empowers organizations to rank their compliance requirements; for example indicate that internal (more strict) auditing requirements take preference over those in relevant legislation like IFRS or that anonymity wishes of customers are honoured unless overridden by legislative requirements mandating explicit identification. Examples of prioritization include the static rule sequence in Prolog and computed rule agenda in OPS5-like rule systems. RBSLA also supports usage of modalities although this is currently limited to alethic and deontic logics. Structuring is provided for in the form of contractual rule sets called modules. The language also provides conceptually natural semantics.

Another body of useful work not explicitly mentioned so far comes from the area of formal logic. However, since many of the works discussed so far are grounded on one form of formal logic or another to give clear and formal semantics to defined rules, we forego on a discussion here; as the features identified so far do not differ from those found in these XML based languages. Categories of formal logic range include predicate, first order and second order logic, modal logics like the aforementioned deontic and temporal logics, and default logics for handling non-monotonicity. They all share the same principle that the world is represented in

terms of facts. This world is then constrained by rules depicting what can and can not be true, which can then be verified through theorem proving (e.g. based on model theory or proof theory). Depending on the particular logic chosen, the rules will be more or less expressive, increasing respectively decreasing the complexity of the theorem proving process.

Lastly, in industry the focus has been mostly on using languages for definition of executable statements and thus languages in this category are not very communicatable and visualizable. These include the first-order-logic based Prolog as well as JSR-94 (<http://www.jcp.org/aboutJava/communityprocess/review/jsr094/>), a JAVA rule engine API implementing part of RuleML. Both of these languages have limited expressive power. Also, their support for modifiability and clustering is very low. Other proprietary approaches have interesting features though, particularly those provided by commercial rule engine companies; such as Blaze Advisor (<http://www.fairisaac.com/Fairisaac/Solutions/Enterprise+Decision+Management/Business+Rules/Blaze+Advisor/>), Haley Inc. (<http://www.haley.com>) and ILOG (<http://www.ilog.com>). These rule system products use explicit constructs to manage the **life cycle** of rules in terms of their status. The applicability of compliance rules can be limited given the period in which a business process takes place, for example IFRS reporting rules for 2006 may not be relevant when creating the yearly financial report in 2007.

They also emphasize the importance of **documentation**, enabling the description of why rules are applied. This is of interest for traceability in compliance improvement, since it allows for example specific compliance rules to be linked to particular compliance legislations. This allows organizations e.g. to stipulate things like 'apply all Sarbanes-Oxley related rules to this business process'. Moreover, when audited organizations can explain their actions by identifying the rules justifying these actions; which in turn will refer to legislative and other forms of regulation through their documentation. Another key feature utilised in commercial rule systems is the assignment of **responsibility**. We already saw in section 2 that this is important to have a clear idea of who is responsible for what in terms of compliance [20] in relation to compliance risk management; which in turn allows delegation of such responsibility along the hierarchical structure of organizations. Lastly, typically such systems have elaborate **versioning** and **history** mechanisms in place to allow management of different variants of the same rule and keep track of changes to rules. These features are of relevance for traceability (and thus compliance improvement) as well.

### 5.2.2. Analysis

Substantial work has also been done in the area of rule **analysis** of rule based systems, which is important for compliance improvement also with regard to consistency of compliance requirements. [PSB92] distinguishes between four types of problem (referred to as anomalies): redundancy, circularity, deficiency and ambivalence. Rules (or parts of rules) contain redundancy if for every possible interpretation of the rules, i.e. in every possible business collaboration design, it does not matter whether they are applied or not. In relation to compliance avoiding redundancy is important, since it simplifies the management of these rules. One dimension is syntactic redundancy such as duplicate rules, rules that are unreachable [BPW02] (i.e. whose condition(s) are never met), rules that are unusable (rules whose conclusion is not used by any other rules), and subsumed rules (in which two rules have the same conclusions, but the conditions of one rule are a subset of those of the other rule). A second category is semantic redundancy in which rules lead to the same conclusion semantically speaking [LTW02]. This can occur in sets of compliance rule if the same concept is expressed in different ways (like cost versus price).

Circularity focuses on problems relating to the occurrence of looping of rules as they are applied. There are several specialized forms of generic circularity like described in [LTW02]. The most basic ones are self-referential rules. These are rules whose consequents are part of their antecedents, e.g. "if A then A". Looping then occurs if there is some way to deduce the first 'A' without having to use the self-referential rule. After that the rule can potentially fire indefinitely, e.g. if 'A' involves an addition operation. Slightly more complex are self-referential rule chains in which there are two or more rules that together constitute a loop. This is particularly a problem when negation is involved, since then it is possible to start with 'A' and deducing 'not A' (this assumes that rule conclusions can contain negated statements) or start with 'not A' and derive 'A'. To avoid this problem stratification can be employed. In stratification cycles that contain negations are identified and remedied, e.g. by creating a dependency (or triggering) graph.

Continuing, a set of rules is deficient if there is a permissible interpretation such that it contains a fact that cannot be deduced from the rules. In relation to business process compliance this would mean that a business process could be defined in a way that it has certain control mechanisms, for which there is no justification. Given the fact that enforcing compliance control requires investments of the organization, this is an undesirable situation. By detecting anomalies like these organizations can be made aware of the problems, and subsequently remedy them; either by providing justification for them or by removing these controls from the business process model. Finally, ambivalence represents a category of anomalies that express that we can deduce conclusions from a rule/chain of rules that are impermissible. This includes both logical (syntactical) inconsistency as well as semantic related inconsistency. Logical inconsistency deals with situations in which two compliance rules lead to opposite conclusions, e.g. one rule stating that a message must be encrypted and another stating that it should not. Logical inconsistency is a special case in which it is not required to make explicit what is permitted or not.

For semantic inconsistencies it is required to explicitly define what is permissible or not. To illustrate, syntactically an authentication and privacy requirement for an activity do not conflict yet from a semantic point of view they do (since authentication requires personal information protected by the privacy requirement). A possible way of specifying such impermissible combinations is through the usage of consistency rules [ORR07]. Identification of inconsistencies is then partially done during design with regard to inconsistencies that will always occur. This is the case if the conclusions of a set of rules will be true under exactly the same circumstances, and these conclusions violate a consistency rule. In the situation that the contradicting rules do not share the same conditions, the inconsistency is potential in nature. Identification of such inconsistencies can only be done through simulations of the business process (via rule extension). This is possible, but computationally expensive [PSB92].

A possible way to resolve inconsistencies is the usage of prioritization as discussed earlier in this section. This is only effective though if the requirements have been specified by the same organization/person, since different organizations can have different preferences. In relation to business process compliance this means that for private processes each organization can simply define its preferences as it likes (naturally conform what is expected from a legislative point of view). In public processes though negotiation with partner organizations might be required to come to a mutual agreement on which compliance requirements override other compliance requirements in case of conflict. Alternatively, in case conflicts arise that can not be automatically resolved user intervention can be employed to provide the preferred course of action manually.

### 5.2.3. Implementation

Lastly, a relevant area is the implementation of the above-discussed issues. Typically, the entities responsible for the application and management of rules are rule engines. Rule engines are software applications that contain definitions of rules, and they are the typical mechanism with which rule specification languages (like the ones analysed above) are implemented [HON05], [RD05]. Referred to by [ROS03] as business logic servers, rule engines control the selection and activation of rules. Sometimes also referred to as an inference engine, a rule engine activates a rule when incoming data matches either its conditions or conclusions. As such, the engine is responsible for taking declarative statements, and automatically ordering, merging and applying them in accordance with a particular reasoning mechanism.

Rule engines are typically classified into production systems and logic systems [HON05]. Production systems are usually data-oriented and are primarily used to manage information. In contrast to these production systems, logic-based systems utilise logic programming for problem solving by inference, e.g. to answer question, infer new knowledge, and so on. Expert systems are an example exponent of these kinds of system. The components of both types of rule engine are by large the same. The difference lies in the fact that production systems contain reactive rules that automatically are applied when conditions are met, while logic based systems involve user initiation to start rule reasoning. There are two types of chaining: 1) backward chaining supports finding proofs of statements based on known facts; and 2) forward chaining allows deducing new information based on known facts. Nowadays, the popular commercial rule engines support both types of reasoning, e.g. the earlier discussed Blaze Advisor, Haley Business Rule Engine and ILOG Business Rule Management System.

In addition to commercial engines a large number of open-source rule engines can be found e.g. for Java (<http://java-source.net/open-source/rule-engines>). One of them is OpenSource (<http://openrules.com/>), which offers a full-scale open source Business Rules Management Framework. OpenSource employs office tools like MS Excel to allow users to define rules. These rules are entered in such tools and automatically translated into executable statements. In the back OpenSource rule specification follows the JSR-94 specification. Rule ranking is supported in the sense that rules are executed in order of definition. Moreover, in case both more generic and precise rules are applicable in the same situation, OpenSource ensures that the more precise one is applied. OpenSource provides some rule verification tools via a OpenRules Plug-in for Eclipse that allows errors in generated programming code to be detected. Mandarax (<http://mandarax.sourceforge.net/>) is another open-source rule engine project (inspired by work in [PAS05] on RBSLA). One part of the project has focused on Prova, a distributed Semantic Web rule engine which supports complex reaction rule-based workflows, rule-based complex event processing, distributed inference services, rule interchange, rule-based decision logic and dynamic access to external data sources, web-based services and Java API. Prova essentially constitutes a merger between declarative Prolog like reasoning with prescriptive Java based programming. The focus is on providing forward and backward reasoning using rules that are declarative specified while referring to Java constructs. Jess (<http://www.jessrules.com>) is a rule engine in Java that uses a text based interface to enable rule specification (using colouring and formatting to facilitate this process). Jess is compatible with the JSR-94 specification. A well known backward reasoning rule engine is Prolog.

The rule engines discussed so far are capable of handling rules with limited complexity. More complex in nature is the Drools rule engine (<http://www.jboss.org/drools/>), an open source rule engine developed by JBoss. Like the other engines, Drools provides an implementation

of the RETE algorithm to facilitate rule reasoning. In addition it also supports expressive power features such as parameter operators, conflict resolution (based on priorities) and reasoning with non-monotonicity, and negation and customizable quantification via cardinalities. From a management perspective Drools enables among others rule versioning, definition of activation and agenda groups to package rules, and activation and expiration dates. For the authoring of rules Drools comes with supported XML formats, templates, a guided editor, decision tables using Excel and OpenOffice, and integration with Eclipse. Rules are specified in DRL, short for the Drools Rule Language, a concise language for rule description (an XML companion is available). Domain specific languages can be defined to present rule definition in more conceptually natural semantics.

A last engine we discuss here is SweetRules (<http://sweetrules.projects.semwebcentral.org/>). SweetRules is an integrated set of tools for semantic web rules and ontologies. It builds on the defacto interoperability standard for rule specification, RuleML, to facilitate rule specification. SweetRules supports the so-called Situated Courteous Logic Programs extension of RuleML, which includes prioritized conflict handling and procedural attachments for actions and tests. SweetRules is capable of translation and interoperability between a variety of rule and ontology languages (including XSB Prolog, Jess production rules, HP Jena-2, and IBM CommonRules), scaleable backward and forward inferencing, and merging of rule bases (e.g. to combine policies). As a final remark, the reader should be aware that in addition to the above discussed tooling options a multitude of other rule engines exist for different languages and implemented for different platforms. As such, the overview provided here should not be considered to be exhaustive but rather is intended to be illustrative in nature.

An interesting aspect of rule engines is that they can be employed within service oriented architectures. Work has been done in this regard for example on the integration of rule engine technology into service enabled business processes through **service orientation**. [NRD06] introduces a service-oriented rule engine that allows organizations to invoke RuleML based service-accessible rule engines. Differently, [KM07] proposes an environment in which rules can be defined and then incorporated into WSDL based web service descriptions, after which these can be deployed in a service execution component. [CM07] describes how to use the earlier discussed technique of aspect orientation to integrate rules into BPEL processes. These rules are captured in aspects, which are weaved into BPEL execution models at runtime. This is similar to what has been suggested in [RD05] in which an ESB is used to coordinate between a BPEL engine and rule engine during execution; allowing the administering of rules to be done by the specialized rule engine component each time a BPEL activity is to be carried out in the BPEL engine. An example of a commercial service oriented rule engine is the Oracle Fusion Middleware Rule engine ([http://www.oracle.com/technology/products/ias/business\\_rules/index.html](http://www.oracle.com/technology/products/ias/business_rules/index.html)). This engine enables ILOG based facts to express rules that are then inserted into BPEL process models.

### 5.3. Policy based approaches

The rule languages discussed in the previous section tend to focus on the specification of individual compliance rules. However, typically people and organizations do not think in terms of specific rules. Rather, as observed in [CHI04] it is usually not possible for us to accurately remember the details of more than a few rules at a time. Therefore, humans prefer to think about collections of rules that together represent some sort of logic that we wish to apply. These rule sets provide organizations with an intuitive logical structure in which they can organize and manage their rules; which in turn can potentially facilitate reuse of rules

across multiple business processes. For these reasons, in the literature substantial work on so-called policy languages can be found.

For example, in the area of service oriented computing we find the already mentioned WS-Policy. WS-Policy provides a general-purpose model and syntax to describe and communicate the policies of a Web service. Assertions can be expressed in the language to reflect provided/requested web service capabilities. These assertions can be attached to web service description constructs in WSDL via WS-PolicyAttachments, e.g. specific messages or operations. WS-Policy itself does not provide any domain specific assertions. Rather the idea is that the language provides basic constructs that are extended in other web service specification; e.g. as done in WS-SecurityPolicy. Policy assertions are relatively basic in nature expressing that a certain capability/requirement must be met. They can be optional in nature, meaning that they may or may not be satisfied. However no conditions for assertions can be stipulated.

Assertions can be clustered though in so-called policy **alternatives**. These alternatives group related sets of assertions. A **policy** then comprises zero or more of such alternatives, which are mutually exclusive (i.e. only one alternative can be applicable at a time). Also, assertions can contain policy constructs in order to allow complex definition, in which a policy contains an assertion, which itself contains a policy refining the assertion. Unfortunately alternatives can not be ordered, as WS-Policy leaves it to other specifications to define such mechanism. Policy expressions can be included through **referencing**, which allows them to be defined once and reused across policies on an as-needed basis. This reuse allows for example to have one policy extend another policy to construct policy hierarchies. It also makes policies more manageable as changes to a policy expression need only be performed in one location.

Related, Web Services Policy Language (WSPL) [AND04] is another language for defining web service policies. Not to be confused with WS-Policy, WSPL is a strict subset of the before discussed eXtensible Access Control Markup Language to describe policies. A policy in WSPL is a sequence of one or more rules. Each policy targets a particular aspect of the web service that is covered by that policy (e.g. a particular operation or message). WSPL policy rules express acceptable choices for satisfying the policy. Rules are listed in order of preference, with the most preferred choice listed first. A WSPL rule is a sequence of predicates. All of the predicates in a rule must be satisfied in order for the rule to be satisfied. Predicates constrain attributes, which are always name-value pairs (as defined in XACML). This allows the use of fine-grained attributes in predicates; where supported operators include equals, greater than, greater than or equal to, less than, less than or equal to, set-equals, and subset. Attributes can be of different XML Schema type (based on those supported by XACML), being string, integer, floating point number (double), date, time, Boolean, URI, hexBinary, base64-Binary, dayTimeDuration, yearMonthDuration, x500-Name, and rfc822Name.

The policies for all aspects of a web service are collected into a so-called PolicySet. Like a Policy, a PolicySet has a target, but here the target specifies the service identifier and the service port type. If a service has different policies for different operations or messages supported by the service, a second level of PolicySets can be nested inside the top, service level, PolicySet. The target for each second level PolicySet specifies the operation, and optionally the message, to which the policies within the PolicySet apply. As said, WSPL is a strict subset of XACML. XACML is an access control markup language that provides a language to define access control policies with. XACML defines three top-level policy elements, being rules, policies and PolicySets. A rule constitutes a Boolean expression comprised of one or more predicates. Conditions define what must be true in order for an

effect to occur (i.e. a conclusion to be true). Rules can be evaluated in isolation. However, they are not intended to form the sole basis for an authorization decision. Rather, it is the basic unit of management that can then be re-used across multiple policies. A policy thus is a collection of rules that together mandate the outcome of an authorization decision according to a defined procedure. Examples of procedures are deny-overrides and permit-overrides, which respectively state that if one rule is (not) met, authorization must (not) be granted.

A PolicySet then contains a set of policies (or other PolicySets) as well as a specified procedure for combining the results of their evaluation. The procedure defined for this is the "Only-one applicable" policy-combining algorithm, which ensures that one and only one policy or policy set is applicable by virtue of their targets. If no policy is applicable, then the result is 'not applicable'. If multiple policies are applicable, then the result is "Indeterminate". When exactly one policy is applicable, the result of the combining algorithm is the result of evaluating the single applicable policy. In this sense policies function somewhat as policy alternatives in WS-Policy depicting a particular course of action. The procedure in the PolicySet then stipulates what alternative is preferred to base this action on (in the case of XACML whether or not to grant authorization). Like for alternatives in WS-Policy though, there is thus no explicit means of distinguishing between situations in which multiple policies are applicable in XACML. [KF07] proposes a deontic logic based policy language for specification of authorization and privacy policies for semantic web services. In these policies constraints specify conditions that must be true at the time of the invocation (concerning e.g. the actor, action or any other relevant construct used to describe the service behaviour). Provisions then depict conditions that must be true after the invocation. This is very similar to for example pre-conditions and post-conditions in OCL.

Rewerse [BO05] is a policy language designed for specification of a variety of different policies, including privacy and security policies. A policy in Rewerse is a set of monotonic rules with non-negated conclusions. The semantics of these rules are therefore equal to stratified programs whose semantics can be interpreted via negation-as-failure in terms of perfect model, stable model or well-founded semantics [GRS91]. Ponder [DDL+00] is another policy language for specification of security policies (as well as so-called management policies). In addition to the features distinguished so far, Ponder allows **grouping** of policies in relation to positions in the organization (called roles). This allows organizations to more easily manage their policies as they are directly related to roles. Moreover, policies can then be structured conform the organizational structure, providing an intuitive way of relating policies. This is made more concrete in [BLR+05], which discusses policy **refinement** to help structure policies in the context of quality of service. The paper describes how policies for QoS can be refined from abstract to concrete requirements using goal oriented techniques. This is of interest for compliance improvement as it helps bridge the gap between the (often) abstract nature of compliance legislations and the low level details required for actual enforcement purposes during business process execution. We return to this matter in more detail in section 5.3.

The Service Component Architecture Policy (SCAP) framework also incorporates the idea of refinement. SCAP defines a framework for associating policies and policy assertions (based on WS-Policy) with service components (as defined in the Service Component Architecture (SCA)). It provides the interesting notion of intents, which comprise abstract requirements for aspects of a service component. For example, in SCA an intent can be that authentication for access to an operation is required. This intent is declarative in nature foregoing specification of how authentication is to be done at runtime. As such, intents can be seen as making explicit choices during design time, which can then be enforced in different ways during execution. Multiple intents can be attached to a single target service construct. However, the work

observes that care must be taken that these do not **conflict**. The SCAP framework leaves open how to detect and resolve such conflicts. For compliance improvement this is important as compliance requirements for business processes can contradict one another (e.g. privacy versus security), and it is necessary to know to which requirement the business process should be compliant (as one of the conflicting requirements will be violated ultimately).

For this reason policy **analysis** has received considerable attention in the literature. [JSS97] notes for example that policy conflicts are either static or dynamic; just like rule conflicts. Static conflicts are those that can be detected at design time and always occur; regardless of the state during execution. Dynamic conflicts are those that occur at run-time and arise because a particular state of the process. These are harder to detect in advance given that it is necessary to analyse the process in all possible states to do so (i.e. through simulation). Policy consistency analysis typically builds on rule analysis in which groups of rules are considered to be consistent if their rules are consistent. The WS-Policy framework informally describes this procedure for examining the consistency of two WS-Policies, while [AND04] and the XACML specification do the same for XACML based policies. In all cases consistency is carried out in a cross product manner in which each alternative in one policy is compared to every alternative in the other policy. If no consistent alternatives are found, the policies are inconsistent. Similar work to this extent can also be found in [ORR07]. Inconsistency handling can be based on rule prioritization and potentially also through ranking of alternatives.

Tooling for the commercially oriented policy languages is also available. For example, NetBeans provides support for the specification of WS-Policy definitions through the Web Services Interoperability Technology (WSIT) Module (<http://websvc.netbeans.org/wsit/>). Though NetBeans does not provide an explicit WS-Policy API, it allows policies to be configured for web services and then included within their WSDL descriptions (or in a separate configuration file). These are then deployed together at runtime. Similarly, the Apache Axis2/C project (<http://ws.apache.org/axis2/c/index.html>) has an implementation including a WS-Policy implementation called Neethi/C (with WS-SecurityPolicy extension). Microsoft provides support for WS-Policy in its set of Web Service Enhancements (WSE, <http://www.microsoft.com/downloads/details.aspx?familyid=fc5f06c5-821f-41d3-4fe-6c7b56423841&displaylang=en>) in the .Net environment through the usage of visual tools. Settings defined by developers in these tools are automatically translated into WS-Policy expressions. For XACML tools have also been developed e.g. the open source JAVA implementation provided by Sun (<http://sunxacml.sourceforge.net/>) and Apache (<http://sourceforge.net/projects/xacmlight/>). Google is also in the progress of developing a JAVA implementation (<http://code.google.com/p/enterprise-java-xacml/>). Work has also been done on XAMCL implementation for .Net such as MVPOS (<http://mvpos.sourceforge.net/>).

## 5.4. Goal based approaches

A third class of approaches relevant for compliance specification comes from the area of goal based development. As the discussion in section 2 made clear, goal oriented compliance is aimed at translating high-level compliance objectives into detailed requirements enforceable during execution. Examples of such approaches included [BAS06] and [GAP07], which link specific legislations to goals that were next refined into more concrete sub goals. The interesting aspect of this for compliance is that such refinement would create an explicit and thus traceable link between business process controls and abstract legislative compliance statements (which is useful for compliance improvement). In the following the area of goal-based development is explored. It would go too far to give a detail overview of the field.

Rather, the focus is on identifying interesting features and characteristics for compliance requirement specification; where the goal overviews are based on [LAM01] and [TKC+06].

Informally speaking, a goal can be considered to be an objective the system under consideration should achieve. Thus, from a goal perspective compliance goal formulations refer to intended compliance properties of business processes that are to be ensured. Goals can cover different types of concern, ranging from functional requirements to non-functional ones. In this sense goals are agnostic to the actual requirements that they express (just like rules). An important purpose behind goal specification is that it helps to contribute to a complete view of system requirements [YUE87]. For business process compliance this is relevant in the sense that organizations can ensure that their compliance goals are complete in relation the stipulated legislative requirements through analysis. This is useful to assure that the results of the compliance discovery process are fully taken into account during compliance specification. Related to this is the avoidance of irrelevant goals [YUE87] (similar to rule analysis to avoid rule redundancy).

Goals may also be formulated at different levels of abstraction. This is of interest for compliance improvement, since often legislative compliance statements are very abstract in nature. By elicitation of goals at different levels of abstraction they can be more easily communicated to the different stake holders in the system. This will allow business managers to express and discuss business process compliance in an abstract manner, while at the same time the IT developers are able to do the same in relation to the achievement of these compliance goals in the automated parts of processes. Moreover, goal **refinement** can then be employed to establish a hierarchical three of high level and low level goals [DLF93]. This creates traceable paths between goals, expressing how higher level goals are achieved through the realization of lower level sub-goals. Besides strict goal refinement work has also been done on linking goals that positively/negatively contribute to other goals. Positive relations are not unlike the chaining of rules via their conditions and conclusions.

Distinguishing between goals at different levels also helps separate stable from more volatile requirements [AMP94]. High-level goals tend to be more stable in nature while the manner in which they are achieved via sub goals is more prone to change. When defining compliance goals based on compliance legislations, the high level goals will not change often (e.g. having accurate financial information). How organizations achieve this, is dependent on their particular internal structure, preferences, processes, and etceteras. Moreover, they may wish to consider multiple alternatives for how to achieve completeness and accuracy, which are then to be used in different circumstances e.g. depending on the crucial nature of the specific piece of financial information. [LAM01] mentions how this can be facilitated through alternative goal refinements; essentially establishing multiple goal trees. An interesting notion in this regard is made by [MCN92], distinguishing between 'soft' and 'hard' goals. Hard goals are goals whose achievement can be directly measured, while soft goal satisfaction cannot be so easily established. High-level goals are typically soft, while underlying sub goals are hard in nature. During alternative goal refinement selection it can be assessed which hard goals contribute 'best' to realizing a soft goal.

Goals can be delegated to actors in the system to allocate **responsibility** of their achievement [LAM01], a useful feature for compliance risk management. Actors can be both humans as well as computerized systems. They can also be passive or active [YUE87]. Achievement of goals may depend on the collaboration of multiple actors [DLF93]. This fits naturally with the idea that realization of compliance of a business process to e.g. a compliance legislation involves many actors; not only in terms of high level versus lower level goals, but also in relation to the fact that one legislation typically addresses a wide variety of different

compliance concerns (as we saw in the analysis of compliance legislations in section 3). This is not unlike the approach taken in [BAS06] in which goals are both refined and allocated to actors to establish an explicit linkage between the realization of compliance goals and the organizational structure of organizations. As such, there is a clear view on which actor is responsible for meeting which goals, how these contribute to higher level goals, and how they are realized in terms of lower level goals (delegated themselves to the same or other actors).

Making goals explicit is also of interest, because of the potential for goal **consistency analysis**. Compliance requirements flowing from legislation and regulation (like in other requirements) often contradict one another. For example, the need for security via authentication conflicts with the preservation of privacy. Such contradictions should be detected and resolved. [LAM01] observes in this regard that this not only encompasses logical inconsistency, but also so-called divergence. Divergence is similar to semantic inconsistency of rules, and concerns goal conflicts like the one between security and privacy. These conflicts can not be identified purely on a logical basis. Rather, auxiliary rules are necessary to detect them. [LAM01] refers to these rules as boundary conditions that (like consistency rules) define what goal combinations are permissible and which are not. Through addition of prioritization statements conflicts can then be addressed by indicating which goal is more important to achieve (e.g. using a priority [DLF93]).

Popular approaches for the specification of goals include KAOS [DFL93], NFR framework [MCN92], and *i\**-models [YU97]. KAOS stands for Knowledge Acquisition in autOmedated Specification. KAOS builds on a conceptual meta-model for acquiring and structuring requirements models (with an associated acquisition language). Meta-model instances constitute graphs in which each node captures a concept such as, e.g., goal, action, agent, entity, or event. The edges between these concepts capture semantic links. Goals can be hard or soft in nature. KAOS comes with a complete formalization using a real-time temporal logic, which allows powerful analysis. The definition of goals in the Non-Functional Requirements (NFR) framework [MCN92] is semi-formal in nature, while [DDM+98] discusses usage of textual versus graphical goal representations. Lastly, *i\**-models in [YU97] provides a formal mechanism for modelling actors and their goals as well as a graphical representation. The *i\** approach has recently been combined with work on e-service values in relation to web services [RGY05].

Having the means to specify compliance goals is important. However, by itself it is not enough for organizations to actually define these goals. For this some form of **elicitation** procedure is needed to facilitate compliance specification. [YKC+06] notes in this regard that why and how questions are typically of usage for this purpose, for example found in [BA05]. [YKC+06] also mentions several works from industry that have given serious thought to goal elicitation. One is called B-SCP [BCV06], which uses interviewing and document elicitation questions to gather goals. These are then structured into a hierarchical model conform the standard model for business rules motivation [KHE+00]. Differently, FBCM [KMK+07] builds on field observation cards to gather requirements from document analysis, workplace observation and interviews. These **interactive** techniques may be of relevance for the identification of compliance goals defined in compliance legislations (particularly document analysis and interviews with legal experts). Differently, works like [DLF93] have developed goal decomposition patterns to help facilitate goal specification.

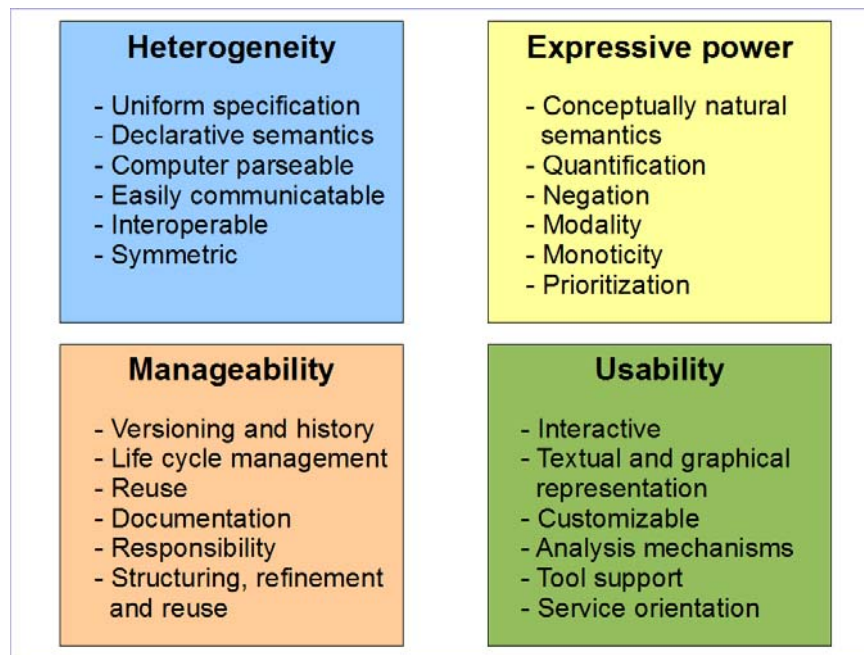
Finally, a very different application of goal-oriented work can be found in constraint satisfaction. Constraint satisfaction is the process of finding a solution to a set of constraints [APT03], essentially expressing goals to be met. An example work following this approach is described in [PAP+02] in which users can encode requests as constraints based upon which a

service composition plan is generated (where the user request language is inspired by the goal language Eagle). Such constraints enumerate the possible values a set of variables may take. A solution is then an evaluation of the variables such that all constraints are satisfied. Such solutions can be automatically found by solving the constraint satisfaction problem. Potentially multiple solutions can be found. This is of use in relation to business process compliance to find multiple compliant business process models based on a set of compliance constraints. Organizations can then decide which model is employed depending on additional preferences. This is different from earlier discussed goal oriented approaches in the sense that there goals are defined in a relatively high level manner as part of the process of requirements analysis for business process design. In contrast, in constraint satisfaction goals are very concrete in nature driving the identification of business process models that satisfy the stipulated goals.

## 5.5. Conclusions

In the previous sections an extensive body of work in the areas of rule, policy and goal languages was analysed in order to identify useful features for specification of compliance rules and policies (both for the definition of compliance requests as well as provisions compliance capabilities). These features fall into four categories: heterogeneity, expressive power, manageability, and usability (see also Figure 3 on the next page). Heterogeneity deals with the fact that compliance policies and rules must be able to address a wide variety of compliance concerns. This implies that specification of such policies and rules is done in an uniform and generic manner. This in turn suggests that the specification language adopted within the COMPAS project will be layered on top of the generic core views within the Model Driven Integration Architecture for Compliance. This will ensure that the specification language can be used to define policies and rules over any of the business process views refining these core views. At the same time the resulting policies and rules will refer to the concepts within developed DSLs for individual compliance concerns, as such providing conceptually natural semantics to compliance stake holders. The easily computer parse-able and communicatable features point towards adoption of an XML based language, while the interoperability feature emphasizes the usage of standards in that regard. Finally, symmetry of the language will promote widespread usage among both service providers and requesters.

The expressive power category contains the potentially interesting features for specification of compliance policies and rules, e.g. to allow prioritization of compliance requirements of relevance in relation to consistency checking and resolution in the context of compliance improvement. Others such as quantification and negation directly impact the support provided for compliance specification. An important consideration from a usability stance in this regard is that the specification language should be customizable. This will ensure that organizations can define their compliance policies and rules at just the desired level of complexity. The features in the manageability category help in some way with compliance improvement and/or compliance risk management. For example, documentation can help link compliance rules to legislations, while delegation of responsibility contributes to compliance risk management. Structuring, refinement and reuse are important for compliance improvement, as they make it easier for organizations to adjust and maintain their compliance policies and rules (e.g. also to include new policies and/or rules in the context of compliance discovery).



**Figure 3: Interesting Features for Compliance Requirement Specification**

Lastly, the usability features deal with the ease with which the specification language can be utilised by organizations. Part of this concerns the representation of language constructs to best facilitate compliance specification. Given the complex nature of compliance it will be crucial that the specification language is intuitive and easy to grasp for non-technical compliance stake holders. In part this is also related to the provided analysis tools to support compliance improvement. This will be vital both in terms of support for testing deviation of business processes to compliance policies and rules as well as analysis of consistency (and resolution of inconsistencies) among compliance policies and rules. Finally, tool support and service orientation are important from a practical point of view to ensure that the adopted specification language is easily implementable and integrate-able with other compliance components (such as the Model Driven Architecture for Compliance or formal analysis component developed within the COMPAS project).

Finally, there exists a potential synergy among the notions of rules, policies, and goals. Specifically, goals can be utilised to capture high level, abstract objectives that are reached through the application of policies; which themselves comprise of sets of concrete rules. In context of the compliance request language this offers the possibility to have compliance goals capture abstract compliance requirements for business processes. These goals would refer to specific compliance legislations and process characteristics and as such be high level in nature. These goals could then lead to the identification of compliance policies containing applicable compliance rules. Such identification process could then be fueled by appropriately annotating compliance policies and rules with statements, which identify to what process(es) a policy and its rules are applicable and what compliance legislations they originate from and enforce. Such approach would enable business managers to pose high level, abstract compliance requests in terms of legislative and process characteristics. These would then lead to the identification of compliance policies and rules defined by compliance experts for enforcing the selected legislations in the identified business processes (or fragments thereof).

## 6. Overall Conclusions

This deliverable has presented the state-of-the-art in compliance languages and approaches, especially focusing on addressing requirements found in compliance legislations. Firstly, existing solutions in different areas of compliance were surveyed in section 2. This resulted in an overview of the current body of work for compliance discovery, specification, reporting, improvement and risk management. It also led to the identification of two gaps in existing works with regard to dynamic compliance checking and compliance enforcement. Given the purpose and scope of this deliverable in relation to its place within the COMPAS life cycle, the focus of the remainder of the deliverable was then on the discovery, specification and improvement aspects of compliance.

With regard to compliance discovery, in section 3 a wide variety of compliance legislations were examined; specifically Basel II, FINRA, HIPAA, IFRS, MIFID, Sarbanes-Oxley and Tabaksblatt (also in conformance with the Description of Work [DOW] for the COMPAS project). From this discussion a set of common compliance concerns was extrapolated, related to control flow, information usage, locative issues, resource employment, temporal requirements, monitoring, payment, privacy, quality, retention, security, and transaction. Example compliance issues in these different areas were identified, also with the help of relevant works such as COBIT and COSO. The lesson learned from this undertaking was that compliance requirements are diverse and cross-cutting in nature and are often defined in the abstract. Interestingly, these concerns can be thought of constituting the core building blocks for definition of higher level compliance concerns. For example, an organization may for example have an auditing compliance concern, which could comprise several of the discussed core compliance concerns, such as secure retention of information and events, but also monitoring of these events. This is an interesting avenue of research not yet explored in current compliance research.

Relevant work for the specification of compliance requirements in the areas covered by these compliance concerns was then surveyed in section 4. The emphasis here was on the identification of distinguishable properties of business processes that can be reused from existing solutions to help cater for explicitly capturing compliance requirements. To this end first a variety of business process modelling solutions was reviewed; both aimed at modelling business processes in general as well as specific dimensions thereof. As became apparent current industry standards such as WS-BPEL and BPMN are capable of addressing some, but not all, of the basic compliance concerns (most notably regarding locative and temporal requirements); and as such may require extension in some form. Next, works were reviewed per advanced compliance concern, i.e. in the area of monitoring, payment, privacy, quality, retention, security and transaction. These works can potentially function as a solid foundation for the development of domain specific languages (DSLs) for those concerns. Several interesting solutions were also found to integrate the concepts within such DSLs into more basic business process models, e.g. via the usage of aspect oriented business process modelling. An open issue remains how to combine constructs from different DSLs, e.g. to express statements such as that events must be securely retained or that authentication events must be logged.

After that, section 5 continued the discussion on compliance specification of section 4, with an emphasis on the expression of compliance policies and rules. This distinction was based on the general approach taken in current compliance proposals to separate the specification of compliance enriched business processes and the actual compliance requirements, which advocate: 1) the need for models to describe business processes including compliance-specific

issues such as retention; 2) the need for models (i.e. policies and rules) to describe the compliance requirements to which business processes must conform; and 3) the explicit separation of these two types of model while at the same time establishing how they are related.

This approach seems a prudent one to adopt within the COMPAS project given the advantages for compliance improvement. Particularly, the externalization of compliance requirements has the potential of catering for linking these requirements to particular compliance regulations. Furthermore, it makes these requirements available for review and analysis with regard to consistency. Moreover, compliance requirements are typically only applicable contingent on particular circumstances. By making them explicit such contingencies can be expressed and taken into account when assuring compliance of business processes. Finally, explicit separation allows the relation between a business process and its compliance requirements to be scrutinized for deviations.

For these reasons a variety of compliance requirement specification languages were examined in section 5 (including works from the fields of policy, rule and goal description). During this examination useful features of proposed approaches for compliance were identified. This resulted in an extensive list of desirable features divided into the categories of heterogeneity, expressive power, manageability, and usability. These will be used as a point of reference during the development of the compliance request language; which is being developed within the COMPAS project in order to assist organizations with the specification of: 1) compliance requests that will help them find and/or generate compliant business processes (or fragments thereof); and 2) compliance policies that will allow them to express compliance related constraints on services used within processes as well as on the process as a whole. Also, in context of the compliance request language the combined usage of goals, policies and rules offers the possibility to have compliance goals capture abstract compliance requirements for business processes. These goals will refer to specific compliance legislations and process characteristics and as such be high level in nature. These goals will then lead to the identification of compliance policies containing applicable compliance rules. This allows business managers to express high level compliance requests that can then be semi-automatically used to deduce which compliance policies and rules are applicable to a business process (which can then be used to find and/or generate compliant business processes (or fragments thereof)).