International Journal of Semantic Computing

ONTOLOGY-BASED INFORMATION FLOW CONTROL OF NETWORK-LEVEL INTERNET COMMUNICATION

ANDREAS KASTEN
Institute for IS Research, University of Koblenz
Koblenz, Germany
andreas.kasten@uni-koblenz.de

ANSGAR SCHERP
Kiel University and Leibniz Information Centre for Economics,
Kiel, Germany
asc@informatik.uni-kiel.de

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

Information flow control on the Internet is a desirable feature when it comes to content such as neo-Nazi propaganda, child pornography, or material showing extreme violence or crimes. In order to provide for a flexible control of information flow on the Internet, we present the pattern system InFO (short for: Information Flow Ontology). InFO provides a common support for different enforcing systems such as routers, proxies, or name servers by abstracting from existing as well as possible future regulation types. Thus, unlike existing solutions, InFO provides information flow control on the Internet-layer, transport-layer, as well as application-layer. In addition, InFO allows for linking the technical implementation of a flow control policy with a human-readable representation including its legal background (law) and organizational motivation (code of conduct). Besides a detailed description of the pattern system, we also provide various examples demonstrating the practical applicability of InFO. InFO has been implemented for name servers, routers, as well as application-level proxy servers. Its source code is available to the public.

Keywords: Information Flow Control; Policy; Pattern System; Core Ontology; Ontology Design Pattern

1. Introduction

In principle, anyone can access the content on the Internet from anywhere. In some cases it is desirable to control the information flow and limit "the actions or operations that a user can perform" [1]. For example, Germany regulates accessing neo-Nazi material due to its history [2]. Other content such as child abuse images is prohibited and considered illegal in almost any country. Besides these examples, there are many other cases where such regulations are desired or even needed [3]. However, a comprehensive approach which addresses the issues related to information flow control on the Internet does not exist yet.

Existing solutions like traditional access control [4, 5] determine who can access the content from where, how often, and under which costs. Access to the content is granted by the content provider via user authentication [1]. Thus, access control relies on explicit user accounts which are known prior to the regulating system. However, this does not address the scenario of regulating access to particular Internet content where users are anonymous and/or the content providers have no interest in restricting access to its content such as in the case of neo-Nazi propaganda. Access control can only be implemented at the content providing nodes. It does not foresee the control of information flow using intermediary communication nodes such as routers or name servers. However, such a feature is required in order to prevent access to material that is regarded illegal in some jurisdiction such as neo-Nazi propaganda. Another approach is policy-based network management [6, 7, 8] where information flow control is implemented on the Internet- and transport-layer of the Open Systems Interconnection (OSI) model [9] such as routers and switches. However, policy-based network management does not consider information flow control on the application-layer [9] like name servers and application-level proxy servers. In summary, the solutions followed so far do not address the control of information flow on intermediary nodes between arbitrary server and client nodes or they are focused on lower levels of the OSI model.

This paper proposes InFO (short for: Information Flow Ontology) as a unifying approach to information flow control that can be applied on the intermediary nodes between a server and a client. It supports not only on the Internet-layer and transport-layer but also on the application-layer of the OSI model. In addition, InFO provides support for different technical implementations of access control located at different types of enforcing communication nodes, i. e. routers, name servers, and application-level proxy servers [10, 11, 12]. InFO consists of a set of ontology design patterns [13]. Ontology design patterns are similar to design patterns in software engineering [14] and provide a generic modeling solution to a recurring modeling problem [13]. Each pattern of InFO implements a different feature for information flow control on the Internet that distinguishes it from the other patterns. However, the patterns are not just a collection of independent ontology design patterns. Rather, the patterns of InFO are designed to be used together. In software engineering, such a set of related patterns is called a pattern system [15].

In summary, the contributions made in this paper by InFO are:

- Providing common support for different enforcing systems (routers, name servers, and application-level proxy servers) operating on intermediate communication nodes between a server and a client. Thus, there is one model for different technical implementations of flow control on the Internet. In addition, by its pattern-based design InFO is prepared for incorporating possible future regulation types.
- As different existing technical implementations of flow control are represented using the same common language model, they can be compared with each other and checked if they implement the same high-level policy (e. g., law or code of conduct) in the same way. This is an important feature as access providers often interpret laws differently which results in different flow control implementations [16].

- A human-readable representation can be linked with the technical implementation, which allows for simple lookups why and how a specific policy has been implemented.
- A unified approach to information flow control on the Internet that can be applied not only on the Internet-layer and transport-layer but also on the application-layer of the OSI model.

The remainder of this paper is organized as follows: The need for information flow control on the Internet using InFO is motivated in the subsequent section by an example computer network with nodes distributed over several countries. The requirements to InFO are discussed in Section 3. The pattern-based design of InFO is described in Section 4. Its extension towards more specific ontologies for routers, name servers, and application-level proxies is presented in Section 5. To demonstrate the practical applicability of InFO, a prototypical implementation and application of InFO for a name server, a router, and an application-level proxy is described in Section 6. The related work in the areas of access control, flow control, and usage control is extensively discussed and compared with InFO in Section 7, before the paper is concluded.

2. Example Computer Network

Computer networks consist of communication end nodes such as servers and clients as well as intermediary communication nodes like routers and application-level proxy servers. An example network connecting various communication nodes located in the USA, Germany, and Saudi Arabia is depicted in Fig. 1. Each country has its own national network which includes smaller subnetworks such as the access provider (AP) networks. AP networks again contain several routers as well as name servers and possibly also application-level proxy servers. End users access the Internet through their AP network in their country. For example, the US client resides within the USA while the DE client is located in Germany. The example network consists of several web servers, each operated by a content provider. The web servers can be accessed by any user from any country. Regulations of information flow in the network can in principle be implemented on routers, name servers, and application-level proxy servers [10, 11, 12]. Routers are able to regulate communication by dropping IP packets. Application-level proxy servers can control the flow of information by filtering the accessed URLs and evaluating the content of web-pages. Name servers can restrict the access to a web server by returning a wrong IP address or none at all. Subsequently, the example network depicted in Fig. 1 is described in more detail.

A communication channel can be established between every user of any country and every web server. The Internet access providers of the users are able to regulate such an establishment [16], since they operate in the same country as their users reside in. They are not only familiar with the laws that the users must abide by law and are required to enforce them **(i)**. Although they are required to enforce the same laws **(ii)**, access providers often interpret these laws differently which results in different flow control implementations [16]. Even if the access providers reside in different countries, they often have to implement transnational laws such as EU directives. Depending on the country where the

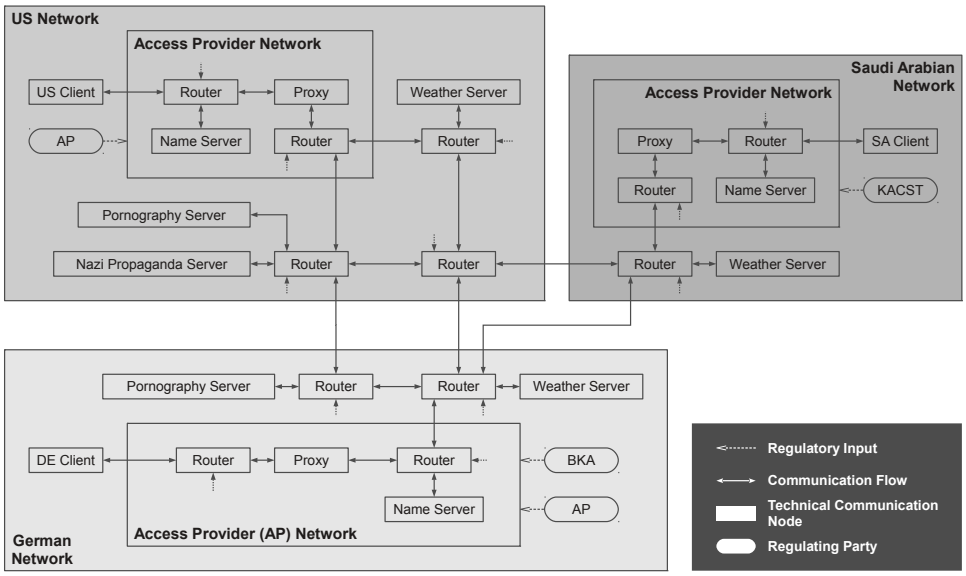


Fig. 1. Illustrating example of a network topology and its involved parties

user lives in, access to particular websites may be either legal or illegal **(iii)**. For example, pornographic content may be legally accessible by German and US users over a specific age[a], but not by Saudi Arabians. Access to neo-Nazi propaganda is legal in the USA and Saudi Arabia but not in Germany according to §86 of the German Criminal Code [17]. Finally, weather information provided by a weather server can be accessed by users of all three countries **(iv)**.

As outlined above, information flow control can be enforced at three different types of network nodes [16], namely routers, application-level proxy servers, and name servers **(v)**. The example network provides different instances of these enforcing communication nodes. Each node type requires specific content identifiers such as IP addresses, domain names, or URLs **(vi)**. Collecting such identifiers differs from country to country and is either done by the access providers or by third parties which are authorized by the country's government **(vii)**. In Saudi Arabia, all content identifiers are collected and managed centrally by the King Abdulaziz City for Science & Technology (KACST) [18]. The USA does not have such a central institution. Instead, the identifiers of the regulated web content are collected and managed decentrally by private parties such as Internet access providers [19]. In Germany, there is a hybrid situation with central assignments to the Federal Criminal Police Office (Bundeskriminalamt, short: BKA) centrally collecting content identifiers and delivering them to the access providers [20]. In addition, several court decisions have required German access providers to manage content identifiers themselves in order to block

[a]Please note: this excludes of course specific content like child abuse images.

access to particular web servers [16]. In addition to the laws of a country that are to be abided by the citizens and companies **(viii)**, an access provider can also define its own code of conduct or guiding principles for information flow control **(ix)**. An example for such principles is the code of conduct of the German Telecom [21]. It basically states that the internationally operating company abides by the national law of the physical location of its subsidiary. Another example for a code of conduct are the Principles on Freedom of Expression and Privacy [22] issued by the Global Network Initiative (GNI). The GNI consists of large companies of the information and communications technology sector including Google Inc., Microsoft Corporation, and Yahoo! Inc. It aims at providing more transparency in Internet regulations. Since the regulations of Internet communication may be derived from different sources such as laws or codes of conduct, some regulations may conflict with each other **(x)**.

3. Requirements

The main goal of InFO is to define ontology design patterns for describing policies for information flow control on the Internet. These policies cover both a human-readable description of their actual meaning as well as the technical implementation details for enforcing them at a particular communication node. An information flow control policy corresponds to a specific use case and consists of several control rules implementing this use case. Multiple use cases require several policies. This section describes the specific requirements for InFO, which are divided into functional and non-functional requirements [23].

The functional requirements are derived from the example network described in Section 2 and the related work discussed in Section 7. The requirements are linked to corresponding aspects of the example network by using the references **(i)** to **(x)**.

(1) Support for different types of enforcing communication nodes. InFO must be able to describe flow control policies which can be implemented by different intermediate communication nodes on the Internet such as routers **(1a)**, name servers **(1b)**, and application-level proxy servers **(1c)**. The scenario defines a computer network which includes all three types of enforcing nodes (see **(v)**).

(2) Operationalization of policies. The actual interpretation of a control policy by a corresponding enforcing node requires a detailed description of the communication flow that shall be regulated. This description must contain all relevant parameters such as the IP addresses of the communicating parties, the URL of the web content, or the domain name of the web server. InFO must be able to describe such parameters. Each of the enforcing nodes of the scenario requires their own parameters in order to work properly (see **(vi)**).

(3) Modalities of control rules. InFO must be able to describe control rules that either allow **(3a)** or deny **(3b)** a communication flow between two communicating parties. The allowance of a communication channel corresponds to the default behavior and is typically not explicitly specified. However, the denial can generally be implemented in different ways such as redirecting to a different communication party or preventing the establishment of the communication channel. InFO must support such different implementations.

(4) Rule conflict resolution. Conflicts between two control rules occur when a particular flow of communication is allowed by one rule and prohibited by another one. The contradicting rules may appear in the same control policy **(4a)** or in different policies **(4b)**. InFO must provide mechanisms for resolving both types of conflicts. In the scenario, the regulations are derived from different sources which may lead to such conflicts.

(5) Identification of policy parties. InFO must provide information about the parties who are responsible for a particular policy. This includes the party who technically enforces the policy (enforcer) **(5a)**, the party who provides the details for this enforcement (provider) **(5b)**, and the party who legislates the enforcement (legislator) **(5c)**. The scenario includes examples of regulation enforcers **(i)**, i. e. access providers such as the German Telecom. Examples for regulation providers **(vii)** are the BKA and KACST. Examples for regulation legislators **(viii)** are the German states and their federation [24].

(6) Identification and classification of regulated content. Regulating access to web content may cause unwanted side-effects such as over-blocking or under-blocking. Over-blocking affects the access to more content than is actually intended. Under-blocking covers only parts of the content to be regulated. In order to reduce such unwanted side-effects, InFO must allow for identifying the content or its hosting server as precisely as possible **(6a)**. Examples for precise identifiers are the IP address(es) of the hosting web server, the domain name of the website, and the URLs of the content itself.

Access regulation is often based on the content's topic such as online gambling, adult websites, or Nazi propaganda. In order to provide a clear reason for why access to particular web content is regulated, the topic of the content must be identified. InFO must provide means for representing the classification of content **(6b)**. In the scenario, the classification of the provided web content is given by the label of the web server, e. g. the "Weather Server" provides documents about the weather **(iii)**.

(7) User's access location. Each information flow control policy is ultimately based on a set of laws issued and active in a specific country. Since an end user's current location also defines the laws she must abide by, InFO must relate a user to her corresponding location. In the scenario, the countries of the end users are the USA, Germany, and Saudi Arabia **(iv)**.

(8) Background information. The regulation represented by a flow control policy is authorized by a legal foundation and/or motivated an organizational code of conduct. In order to enrich a policy with some human-readable explanations, InFO must be able to attach corresponding background information to the policy in form of its legal justification **(8b)** and/or organizational motivation **(8a)**. As outlined in the scenario, §86 of the German Criminal Code is an example for a legal justification **(ii)** which prohibits the distribution of neo-Nazi material. The code of conduct of the German Telecom is a statement to actually enforce this law **(ix)** as well as the local laws of other countries in which the Internet access provider operates.

Besides the functional requirements, the ontology must also fulfill non-functional requirements as stated in [25].

(9) Modularity. InFO must be able to support different types of enforcing nodes. Each type of nodes requires specific enforcement details, which may be irrelevant for other enforcing nodes. For example, a name server requires domain names and does not process

IP addresses, which are used by routers. In order to allow for a flexible use of the actually needed parts of the ontology, InFO must have a modular design.

(10) Extensibility. Although InFO must natively support three different types of enforcing nodes, this support can only cover a limited set of all possible attributes and functions a specific node can have. For example, the build-in support for routers as enforcing nodes does not guarantee a complete support for all functions of all possible routers like Cisco's 3945 Integrated Services Router[b]. An extensible design allows for further enriching InFO with product-specific concepts. Furthermore, such a design can also be used for defining new concepts for future regulation mechanisms.

4. Information Flow Ontology (InFO)

Having introduced the need for information flow control on the Internet and the requirements to an ontology-based solution to this problem, this section presents the pattern-based design of InFO. InFO is a set of several ontology design patterns [26, 13] for describing flow control policies. Adapted from software engineering, an ontology design pattern provides both a description of a specific, recurring modeling problem that appears in a specific modeling context and presents a proven, generic solution to this problem [15, 14]. The solution consists of a description of the required concepts, their relationships and responsibilities, and the possible collaboration between these concepts [15]. An ontology design pattern is independent of a particular application domain [14] and can be used in a variety of different application contexts. Each pattern of InFO implements a different aspect of the flow control that distinguishes it from the other patterns. The patterns are not a collection of independent ontology design patterns but are instead designed to be used together. Thus, InFO corresponds to what is called a pattern system [15]. The pattern system reuses and extends design patterns from the upper ontology DOLCE+DnS Ultralite (DUL) [27] and the Ontopic core ontology[c]. It is implemented using the Web Ontology Language (OWL) [28] and axiomatized using Description Logics [29]. This section first briefly describes these reused ontologies and gives an overview of the InFO pattern system. Subsequently, each pattern is explained in more detail.

4.1. *Modeling methodology and reused ontologies*

InFO uses DOLCE+DnS Ultralite (DUL) [27] as foundational ontology since it provides a rich axiomatization based on several design patterns. In addition, the use of DUL has proven to be a good design choice [25]. The ontology defines the class `Entity` and its subclasses `Object`, `Quality`, and `Abstract`. `Object`s are entities that exist in time and space such as `Agent`s. A `Quality` describes a feature of an `Entity` whose feature value is specified by a `Region`. `Abstract` refers to entities that do neither have spatial nor temporal features. `Region`s are `Abstract`s and represent data value spaces. The

[b] `http://www.cisco.com/c/en/us/products/routers/3945-integrated-services-router-isr/index.html` (last accessed: 03/11/14)
[c] `http://ontologydesignpatterns.org/ont/dul/ontopic.owl` (accessed: 10/06/2014)

relations between these three classes are covered by DUL's qualities and quality region pattern [27]. Other design patterns used from DUL are the collection pattern, the sequence pattern, the information realization pattern, and the description and situation (DnS) pattern. The collection pattern defines the property `hasMember` which can be used for describing the relationship between a collection and its elements. The sequence pattern defines the property `follows` which allows for ordering entities. The information realization pattern defines the classes `InformationObject` and `InformationRealization` as well as the property `isRealizedBy`. An `InformationObject` is an abstract piece of information and is realized by a physical object or digital resource, which corresponds to an `InformationRealization`. The DnS pattern [30] is a central design pattern of DUL and models n-ary relationships between entities such as `Agents` and other `Objects` within a specific `Situation`. The `Description` of this `Situation` defines the context and functions of all involved entities. Finally, the Ontopic ontology defines the class `Topic` and a corresponding pattern. A `Topic` represents a collection of semantically related `InformationObjects`. An example for `Topic` is "neo-Nazi propaganda" which is basically a collection of all neo-Nazi propaganda material.

4.2. *Overview of the pattern system*

An overview of the InFO pattern system is depicted in Fig. 2. The patterns are subdivided into the Technical Regulation patterns, the Organizational Regulation patterns, and the Legal Regulation patterns. Each category of patterns implements a specific aspect of information flow control. The Technical Regulation patterns cover the description of all technical regulation details which are InFO's main focus. The Organizational Regulation patterns and the Legal Regulation patterns support the technical policies with human-readable descriptions.

In detail, the Technical Regulation consists of five different patterns which are based on the DnS pattern. Each pattern models a different flow control aspect which defines the context of the involved entities. The DnS pattern is used since it allows entities such as policies to participate in a specific context by fulfilling a specific function. Policies are basically descriptions of regulations and are thus modeled as subclass of `Description`. Their implementation leads to a `Situation` where each concept defined by the policy is fulfilled by a corresponding entity. The Flow Control Rule Pattern describes a flow control rule which covers the technical regulation details for a particular communication flow. It describes whether or not this flow of communication is to be allowed or denied and thus implements requirements **(3a)** and **(3b)**. The regulation details cover an identifier **(6a)** and a classification **(6b)** of the content to be regulated as well as the location of the user accessing the content **(7)**. All technical regulation details are provided by an agent who corresponds to the rule provider **(5b)**. The Flow Control Rule Pattern is further extended by the Redirecting Flow Control Rule Pattern and the Replacing Flow Control Rule Pattern, which provide for more complex rules. Several flow control rules sharing the same purpose are combined to a flow control policy, which is provided by the Flow Control Policy Pattern. In order to further describe the rules' purpose, a flow control policy is linked to an organizational code

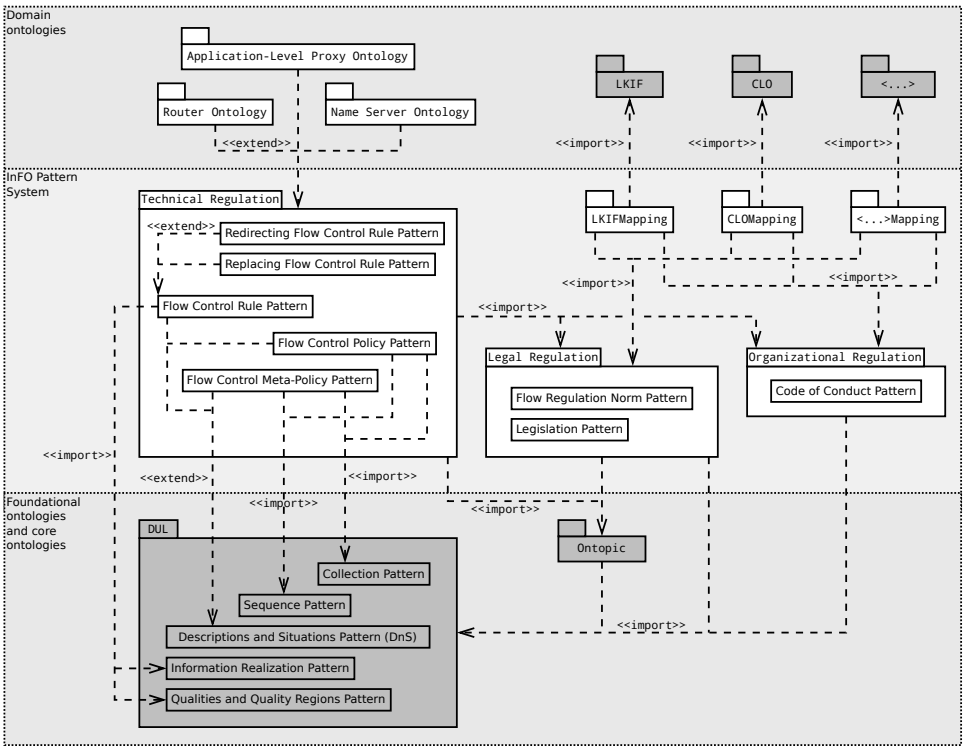


Fig. 2. Overview of the InFO pattern system. Dark gray elements are external ontologies reused by InFO whereas white elements are patterns of InFO or their domain-specific extensions.

of conduct and/or a legal foundation. Therefore, the Flow Control Policy Pattern imports the Organizational Regulation patterns and the Legal Regulation patterns. A flow control policy also covers the enforcing party **(5a)** and the enforcing system in form of routers **(1a)**, proxy servers **(1c)**, and name servers **(1b)** which implement the flow control. Possible conflicts between rules of one **(4a)** or more **(4b)** policies are resolved by using a meta-policy described with the Flow Control Meta-Policy Pattern.

The Organizational Regulation patterns and the Legal Regulation patterns enrich the Technical Regulation with human-readable descriptions. Instead of defining all details for covering such organizational descriptions and legal descriptions in InFO directly, the patterns define concepts which can be integrated into existing ontologies such as the Legal Knowledge Interchange Format (LKIF) [31, 32] or the Core Legal Ontology (CLO) [33, 34]. This allows for re-using all concepts defined in these ontologies together with the Technical Regulation of InFO. An external ontology is integrated by using a corresponding mapping ontology like LKIFMapping and CLOMapping shown in Fig. 2. The Code of Conduct Pattern fulfills requirement **(8a)** by defining concepts to describe an organizational code of conduct as well as its legal background. The Flow Regulation Norm Pattern defines the legality of a particular communication flow **(8b)**. The Legislation Pat-

Table 1. Functional requirements and their implementations by the InFO patterns

	Requirement	Implementation
(1a)	Support for routers as enforcing nodes	Router Ontology
(1b)	Support for name servers as enforcing nodes	Name Server Ontology
(1c)	Support for proxy servers as enforcing nodes	Application-level Proxy Ontology
(2)	Operationalization of control policies	Router Ontology, Application-level Proxy Ontology, Name Server Ontology
(3a) + (3b)	Modalities of control rules	Flow Control Rule Pattern
(4a) + (4b)	Rule conflict resolution	Flow Control Meta-Policy Pattern
(5a)	Identification of regulation enforcer	Flow Control Policy Pattern
(5b)	Identification of regulation provider	Flow Control Rule Pattern
(5c)	Identification of regulation legislator	Legislation Pattern
(6a) + (6b)	Identification and classification of regulated content	Flow Control Rule Pattern
(7)	Access location	Flow Control Rule Pattern
(8a)	Organizational background	Code of Conduct Pattern
(8b)	Legal background	Flow Regulation Norm Pattern

tern pattern allows for modeling how the legal norm conceptualized in the Flow Regulation Norm Pattern was actually created. This corresponds to a legislative procedure and allows for specifying the legislator of the norm **(5c)**.

Policies for particular types of enforcing nodes are described using domain ontologies and are specialized from the Technical Regulation patterns. Policies for IP-based regulation are described using the Router Ontology **(1a)**, policies for the Domain Name System use the Name Server Ontology **(1b)**, and policies for proxy servers are based on the Application-level Proxy Ontology **(1c)**. Each domain ontology provides concepts for precisely specifying all parameters required for implementing the flow control **(2)** in the specific type of enforcing node. For example, the Router Ontology contains concepts for IP addresses, the Name Server Ontology allows for modeling domain names, and the Application-level Proxy Ontology provides concepts for URLs.

The non-functional requirements modularity **(9)** and extensibility **(10)** cannot be mapped to a particular pattern, since they affect the InFO pattern system as a whole. Requirement **(9)** is addressed by InFO's modular design and its use of DUL as modeling basis. Requirement **(10)** is supported by allowing for the creation of new domain ontologies besides the already existing ones.

Overall, we can state that InFO covers all functional as well as non-functional requirements stated in Section 3 (for a detailed discussion please refer to the related work in Section 7). In the following, each pattern of InFO is described in more detail. The descriptions cover a figure for each pattern and a textual explanation. We begin with the Technical Regulation rule patterns, followed by the Flow Control Policy Pattern and the Flow Control Meta-Policy Pattern. Afterwards, the Organizational Regulation patterns and the Legal Regulation patterns are described.

Fig. 3. Basic structure of a communication aspect in the Flow Control Rule Pattern. `<Aspect>` is a placeholder for `Sender`, `Receiver`, `Content`, and `Channel`.

4.3. *Flow Control Rule Pattern*

The Technical Regulation rule patterns cover three different patterns for expressing flow control rules. These patterns are the Flow Control Rule Pattern, the Redirecting Flow Control Rule Pattern, and the Replacing Flow Control Rule Pattern. All three patterns define several communication aspects such as the communicating end points and/or the transmitted content. This is modeled in a common structure which is depicted in Fig. 3. The structure defines a communication aspect as a `Role` which is played by an instance of the class `Object` or one of its subclasses. Example objects are client systems, web servers, or web pages. An object is identified by its features which are described as a quality of the object. According to DUL's qualities and quality region pattern, the actual values of these features are modeled as `Regions`. Possible identifiers for client systems and web servers are IP addresses and domain names, possible identifiers for web pages are URLs, and possible identifiers for communication channels are protocol names such as http or ftp. Each communication aspect is further specified by a corresponding `<Aspect>Specifier` specifier which parametrizes its `Region`. The placeholder `<Aspect>` can be a `Sender`, `Receiver`, `Content`, or `Channel` [35].

A flow control rule regulates if the establishment of a particular Internet communication is to be allowed or denied. The Flow Control Rule Pattern allows to describe such a regulation by associating the regulating rule with four different aspects of an Internet communication. These aspects are defined according to Shannon's communication model [35] and cover the sender and receiver of the communication as well as the transmitted content and the communication channel. All aspects are modeled using the basic structure depicted in Fig. 3. Such a generic solution allows the Flow Control Rule Pattern to cover almost any arbitrary type of information flow. For reasons of simplicity, Fig. 4 only depicts the aspect's role as well as the classified object. If a rule does not specify one of the four aspects, it will be evaluated for all possible aspects. For example, if a rule does not explicitly define a sender, it will be evaluated for all senders. Besides the four communication aspects, the Flow Control Rule Pattern also defines the provider of the rule as well as a regulated topic. `RuleDataProvider` represents the party who creates a flow control rule by providing all information for technically enforcing it. This includes the identifiers of all communication aspects such as IP addresses, domain names, or URLs. Possible `RuleDataProviders` are the BKA and KACST (see scenario in Section 2). The regulated

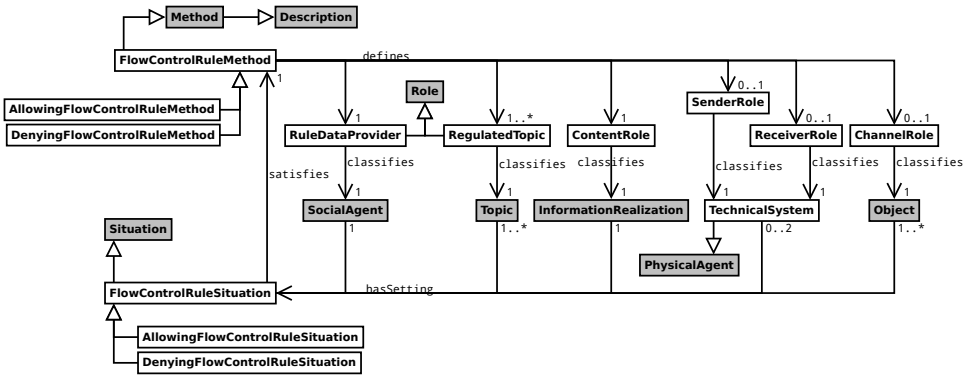


Fig. 4. Flow Control Rule Pattern

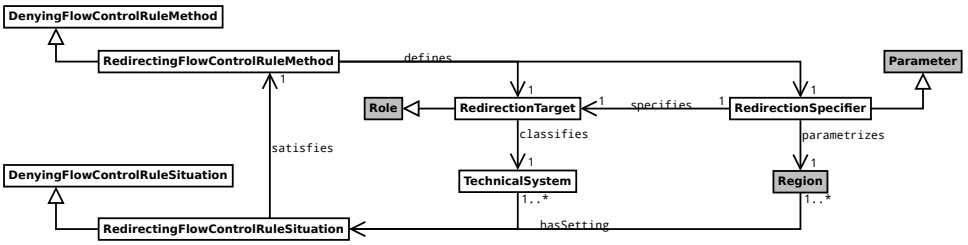


Fig. 5. Redirecting Flow Control Rule Pattern

topic describes the content whose transmission is regulated by the rule. Example topics are neo-Nazi propaganda or pornography.

The main class of the pattern is `FlowControlRuleMethod` which is a subclass of DUL's `Method`. `FlowControlRuleMethod` itself does not specify whether the described flow control shall be allowed or prohibited. This is expressed by its subclasses `AllowingFlowControlRuleMethod` and `DenyingFlowControlRuleMethod`.

The Flow Control Rule Pattern is extended by the Redirecting Flow Control Rule Pattern and the Replacing Flow Control Rule Pattern. The Redirecting Flow Control Rule Pattern allows for denying a particular communication flow by replacing the original receiver with another one. The intended communication flow is therefore not possible. The pattern may be useful if the sender of the communication shall be redirected to a web page which explains the reason for the regulation. The Redirecting Flow Control Rule Pattern is depicted in Fig. 5 and extends the Flow Control Rule Pattern by a `RedirectionTarget` modeled according to the basic structure shown in Fig. 3. The Replacing Flow Control Rule Pattern depicted in Fig. 6 denies a particular communication flow by replacing one of its four basic communication aspects with another one. More specifically, the Replacing Flow Control Rule Pattern replaces the intended content with a replacement content. It extends the Flow Control Rule Pattern by defining a `ReplacementTarget`.

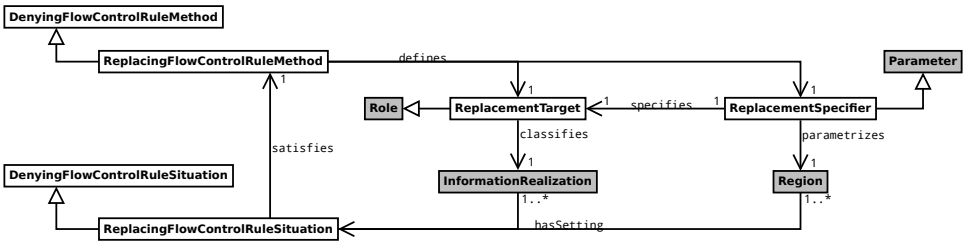


Fig. 6. Replacing Flow Control Rule Pattern

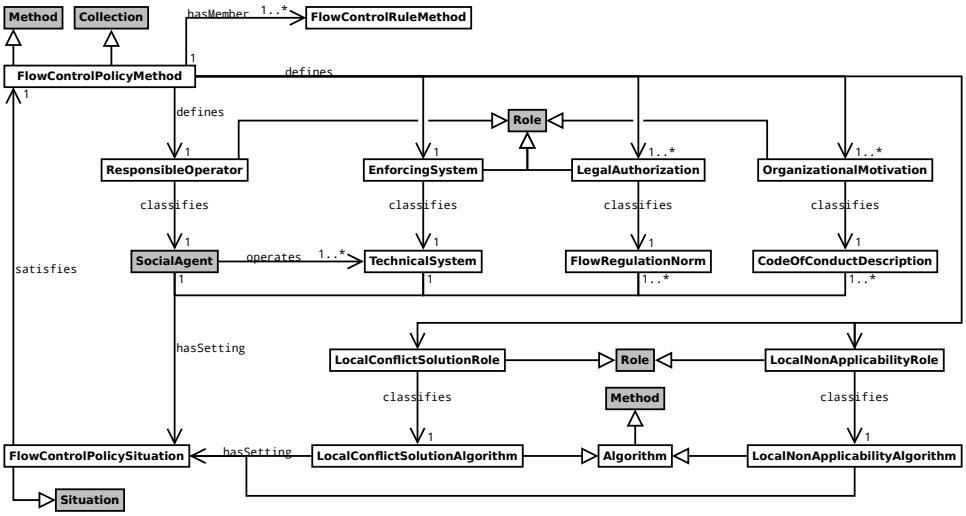


Fig. 7. Flow Control Policy Pattern

4.4. *Flow Control Policy Pattern*

A flow control policy is a collection of multiple flow control rules sharing the same purpose. The Flow Control Policy Pattern depicted in Fig. 7 allows for defining such collections and associates them with a legal norm and/or code of conduct (see Section 4.6). It associates a flow control policy with one enforcing party **(5a)** and one technical enforcing system. The party is represented by an `SocialAgent` and acts as a `Responsible-Operator`. Example operators are natural persons and organizations. The system which technically implements the flow control is modeled as `TechnicalSystem` in the role of an `EnforcingSystem`. Example systems are routers, name servers, and application-level proxy servers [10, 11, 12].

In order to solve conflicting rules, two optional conflict solution algorithms are provided. The `LocalConflictSolutionAlgorithm` defines how conflicts between two contradicting flow control rules of the same policy are resolved. A local conflict occurs when rules of the same policy share the same specifiers of their aspects but differ in the

actual handling of the communication flow. For example, one rule is allowing the specified communication flow while another rule is prohibiting it. Before the algorithm is evaluated, all rules of the policy are ordered according to the rule priority algorithm of the policy's meta-policy which is further described in the next section. If a flow control policy itself does not specify a conflict solution algorithm, conflicts between rules will be solved by the policy's meta-policy. The `LocalNonApplicabilityAlgorithm` covers the handling of such flow control rules which cannot be fully implemented by the enforcing system. An example for such rules is described below in Section 4.5.

4.5. *Flow Control Meta-Policy Pattern*

A flow control meta-policy provides algorithms for resolving conflicts between two contradicting rules of one or more flow control policies. The Flow Control Meta-Policy Pattern depicted in Fig. 8 provides a conceptualization for such a meta-policy. It defines the class `FlowControlMetaPolicy` as collection of several flow control policies and four different conflict solution algorithms. These algorithms are the `PolicyPriority-Algorithm`, the `RulePriorityAlgorithm`, the `GlobalConflictSolution-Algorithm`, and the `GlobalNonApplicabilityAlgorithm`. The algorithms are inspired by the policy languages XACML [4], DEN-ng [6], the Ontology-Based Policy Translator (OPoT) [7], Ponder [36], and ODRL [37, 38] as described in the related work in Section 7. However, InFO provides a more fine-grained and flexible approach for solving conflicts than these policy-languages. Each algorithm covers a specific aspect of the conflict solution process which is further described below. The behavior of a particular algorithm is specified via a corresponding subclass of the algorithm type. For example, possible `GlobalConflictSolutionAlgorithm`s are `IgnoreAffectedRulesAlgorithm` and `IgnoreAffectedPoliciesAlgorithm`. The former algorithm only discards the conflicting rules but leaves other rules of the same policy unchanged. The latter algorithm discards the whole policies of the conflicting rules. Additional algorithms are described in our website, which is referenced in the conclusion.

Furthermore, a flow control meta-policy defines the enforcing system's default behavior via a `DefaultRule`. Each flow control rule covers a specific communication flow. If no rule can be applied to a particular communication, the `DefaultRule` will be used instead. This rule does not define any specific sender, receiver, content, or channel. Instead, it only covers those parameters necessary for the rule's implementation, e. g., redirection targets or replacement targets. A default rule will be evaluated for every communication as long as there is no other flow control rule which already covers that communication. Similar to the Flow Control Policy Pattern, the Flow Control Meta-Policy Pattern associates a meta-policy with one enforcing party and one technical enforcing system. Each enforcing system is related to exactly one flow control meta-policy and can implement multiple flow control policies and corresponding rules.

Lupu and Sloman distinguish between two different categories of possible conflicts of rules which are *modality conflicts* and *application specific conflicts* [39]. Modality conflicts between two rules occur when the establishment of a particular flow of communication

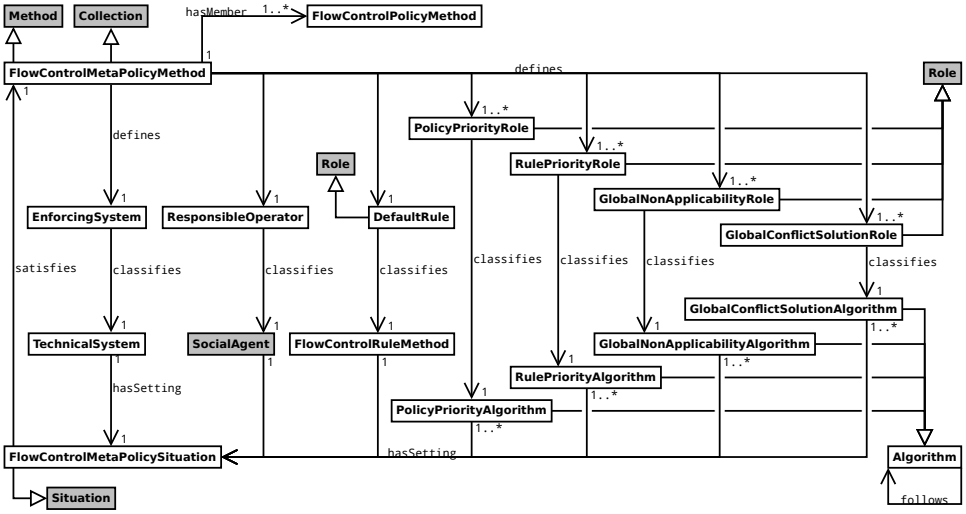


Fig. 8. Flow Control Meta-Policy Pattern

is allowed by one rule and denied by the other rule. Modality conflicts are resolved by the `PolicyPriorityAlgorithm`, the `RulePriorityAlgorithm`, the `GlobalConflictSolutionAlgorithm`, and the (optional) `LocalConflictSolutionAlgorithm`. Application specific conflicts correspond to an incompatibility between a flow control rule and its enforcing system. InFO's open design allows for defining new types of rules by creating a corresponding subclass of `FlowControlRuleMethod`. However, if the enforcing system does not understand this new rule type, it cannot interpret it. Application-specific conflicts are handled by the `GlobalNonApplicabilityAlgorithm` and the (optional) `LocalNonApplicabilityAlgorithm` which are evaluated before applying any other conflict solution algorithm. All algorithms are evaluated according to the following steps: First, the `LocalNonApplicabilityAlgorithm` is applied on the rules of each flow control policy which defines such an algorithm. Second, the `GlobalNonApplicabilityAlgorithm` is applied on the rules of all other flow control policies associated with the meta-policy. Third, all flow control policies are ordered according to the meta-policy's `PolicyPriorityAlgorithm`. Fourth, the rules of each flow control policy are ordered according to the meta-policy's `RulePriorityAlgorithm`. Fifth, the `LocalConflictSolutionAlgorithm` is applied on conflicting rules that are part of the same policy. If a policy does not define a local conflict solution algorithm, this step is skipped. Finally, the `GlobalConflictSolutionAlgorithm` is applied on all other conflicting rules. The first two steps resolve all application-specific conflicts. After these steps, every flow control policy only contains such rules which can be completely interpreted by the enforcing system. Modality conflicts which can be resolved by defining different priorities of the conflicting rules are resolved with the third and the fourth step. Rules with a low priority that are in conflict with a rule of higher priority are

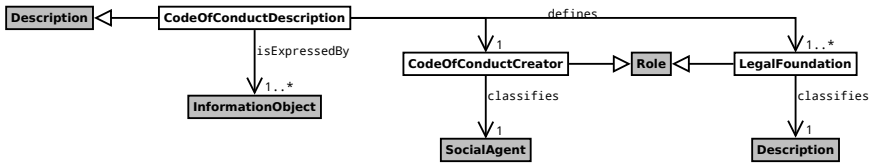


Fig. 9. Code of Conduct Pattern

ignored by the enforcing system. Any modality conflict which still remains after the third and fourth step is resolved during the last two steps. In order to achieve this, the `GlobalConflictSolutionAlgorithms` are designed to remove any conflicting rule or their corresponding policy in the final step. A flow control meta-policy must define at least one algorithm for each type. If there is more than one algorithm per type, the property `follows` from DUL's sequence pattern defines the order of their application. Evaluating the six steps above ensures that all remaining rules can completely be interpreted by the enforcing system. However, if rules or policies are removed during this process, manual intervention is required to further analyze and resolve the cause of the conflict.

4.6. *Organizational Regulation and Legal Regulation Patterns*

The Organizational Regulation patterns and the Legal Regulation cover human-readable background information on the information flow control. The patterns are designed to integrate other, existing ontologies such as LKIF [31, 32] or CLO [33, 34]. This flexible design allows for using different external ontologies with variable expressiveness in different scenarios. The Organizational Regulation defines the Code of Conduct Pattern and the Legal Regulation defines the Flow Regulation Norm pattern and the Legislation Pattern. The following subsections first describe these three patterns.

The Code of Conduct Pattern depicted in Fig. 9 allows for describing the organizational code of conduct on which a technical flow control implementation is based. A code of conduct is represented by the class `CodeOfConductDescription` and is created by the organization acting as the `CodeOfConductCreator`. It is based on at least one legal foundation such as a legal norm or a law. The legal foundation may define the boundaries of a code of conduct stating that the code must not violate any legal norm. A code of conduct is expressed by an `InformationObject` that describes the code in a human-readable form.

The Flow Regulation Norm Pattern depicted in Fig. 10 models the legal state of a particular communication flow. The pattern can be considered as legal view on the rather technical Flow Control Rule Pattern described in Section 4.3. The Flow Regulation Norm Pattern models a particular communication flow as set of events including the communicating parties and the transmitted content as well as the content's topic. Each party is represented by its technical communication system such as a web server or a user client and the specific agent who uses that system. The specific relations between all entities depend on the legal ontology integrated into the Flow Regulation Norm Pattern (see Section 4.7).

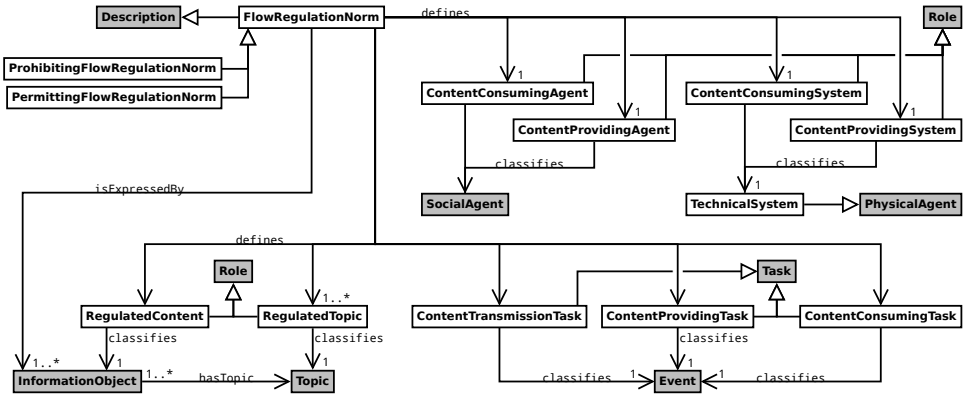


Fig. 10. Flow Regulation Norm Pattern

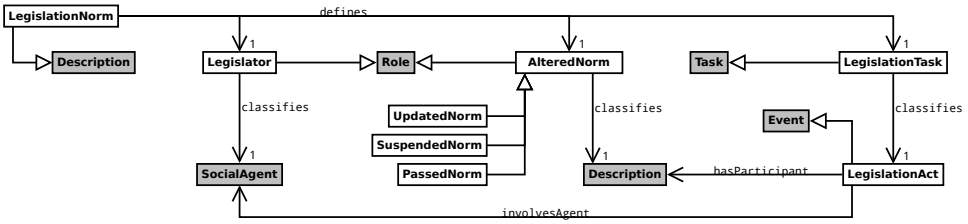


Fig. 11. Legislation Pattern

The Legislation Pattern depicted in Fig. 11 models the process of altering a legal norm. This process is considered a legislation act which is performed by a legislator who is responsible for the process. The Legislation Pattern has a similar design as the Flow Regulation Norm Pattern. Its main concept `LegislationNorm` is associated with all concepts relevant for passing or modifying a legal norm.

4.7. *Integration of Existing Legal Ontologies into InFO*

The Organizational Regulation patterns and the Legal Regulation patterns are designed for using existing legal ontologies together with InFO by integrating them into InFO's pattern system. The main goal of this integration is the re-use of the legal ontologies without modifying or re-factoring them beforehand. Statements of the integrated existing legal ontologies must be valid in both InFO as well as in the original legal ontology. The integration is done in four steps and based on the alignment method by Scherp et al. [40]. In the first step, the structure and design choices of the ontology are investigated. In the second step, existing groups of concepts and properties are identified. The third step corresponds to the actual integration and is based on creating a mapping ontology. As depicted in Fig. 2, a mapping ontology imports the Organizational Regulation patterns and the Legal Regulation patterns of InFO as well as the legal ontology to be integrated. A mapping ontology does

not define any new classes or properties. Instead, it only defines subclass and subproperty relationships between the concepts and properties of InFO and the legal ontology. However, in contrast to Scherp et al. [40], modifying the original ontology by, e. g., removing concepts or axioms is not intended. Therefore, the internal structure of both InFO and the integrated legal ontology remains intact. The fourth step is the validation of the mappings and can be done using an ontology reasoner.

Fig. 2 shows how the legal ontologies LKIF [31, 32] and CLO [33, 34] are integrated into InFO. They can be used for both describing the organizational background and the legal background of an information flow regulation. The actual integration is done with the mapping ontologies `LKIFMapping` and `CLOMapping`. Both mapping ontologies import the Organizational Regulation patterns and the Legal Regulation patterns as well as their respective legal ontology and define additional statements for the integration. The details of the mapping ontologies are beyond the scope of this paper and are available at InFO's website, which is referenced in the conclusion.

4.8. *Summary*

The pattern system InFO consists of several ontology design patterns which cover specific aspects for describing the regulation of information flow on the Internet. These aspects are either of technical, organizational, or legal issue. The main focus of InFO is the technical regulation of Internet communication. The Organizational Regulation and the Legal Regulation are designed to be used together with existing legal ontologies. The subsequent section demonstrates the application of InFO using the example computer network from Section 2.

5. Detailed Example of Applying the InFO Pattern System

A subnetwork of the example network described in Section 2 including more technical details of the regulation's enforcement is depicted in Fig. 12. It is implemented using three different domain ontologies that extend InFO's Technical Regulation patterns. These ontologies are the Router Ontology, the Name Server Ontology, and the Application-Level Proxy Ontology. They extend generic concepts such as `FlowControlRuleMethod`, `Region`, and `Quality` with concepts specifically designed for the technical environment they are applied to. For example, the Router Ontology defines subclasses of `Region` for modeling IP addresses, the Name Server Ontology introduces concepts for expressing domain names, and the Application-Level Proxy Ontology covers classes for modeling URLs. These subclasses allow for a precise definition of flow control rules specifically designed for enforcing systems such as routers, name servers, and proxy servers. This section presents example flow control regulations for each domain ontology.

Access to the computer network `cn-1` shall be prohibited for German users since the network contains several servers hosting neo-Nazi material. The network `cn-1` represents the computer network hosting the website Stormfront, an online platform providing neo-Nazi material [16, 41]. The Stormfront case is used as basis for the example flow control policies given in this section since it corresponds to a real-world sce-

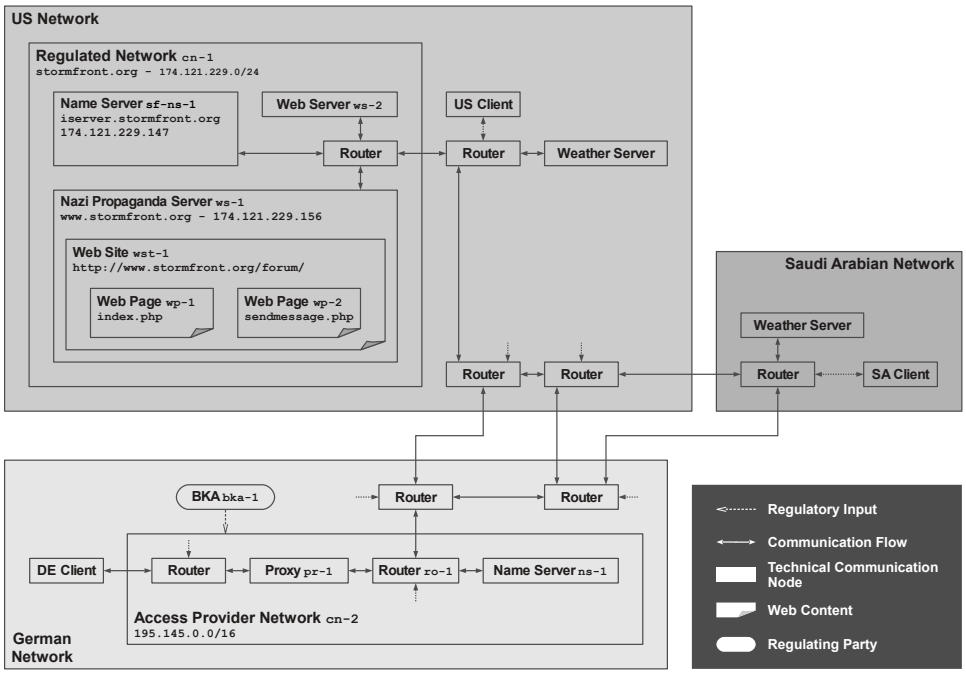


Fig. 12. Regulated web servers and web content of the example policies. The depicted topology is a subnetwork of Fig. 1.

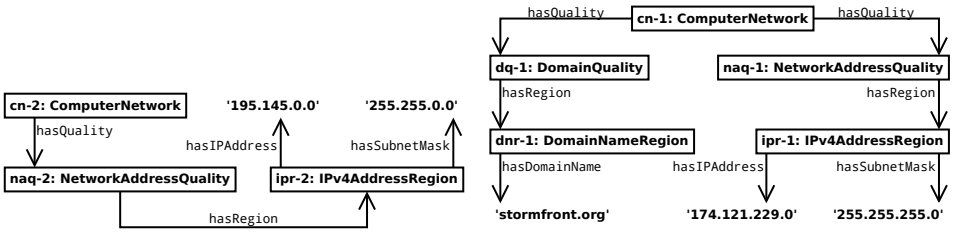
nario. Furthermore, the web site was the target of several Internet regulations in the past and is still regulated in some way, e. g. the French[d] and the German[e] version of the Google search engine exclude the website from its search results [42]. For a detailed discussion of the access regulation of Stormfront, please refer to [16, 41]. The network address of the Stormfront network `cn-1` is `174.121.229.0/24` and its domain is `stormfront.org`. The postfix `/24` of the network address denotes the CIDR notation [43] of the network's subnet mask. `24` corresponds to the subnet mask `255.255.255.0`. The network contains a name server represented by the individual `sf-ns-1` and two web servers represented by the individuals `ws-1` and `ws-2`. While `sf-ns-1` is a regular name server managing the domain names of the domain `stormfront.org`, `ws-1` corresponds to the web server providing the Stormfront web forum. The web server can be accessed by its IP address `174.121.229.156` or by its domain name `www.stormfront.org`. The web forum is identified as the website `wst-1` with the URL `http://stormfront.org/forum/`. The website is basically a set of all web pages with a URL like `http://*.stormfront.org/forum/*` where `*` is a placeholder for arbitrary strings. In particular, the website contains the two web pages

[d] `http://www.google.fr` (last accessed: 03/11/14)
[e] `http://www.google.de` (last accessed: 03/11/14)

`index.php` and `sendmessage.php` which are identified as `wp-1` and `wp-2`, respectively.

The example regulations described below block access to those parts of the computer network `cn-1` that can directly be associated with neo-Nazi material. At the same time, the regulations allow access to other network nodes and content. The name server only provides a mapping between domain names and IP addresses and does not host any web content of neo-Nazi material itself. Similarly, the contact form can also be considered harmless. It provides a communication method similar to emailing, which was not explicitly covered by governmental blocking orders [16]. For illustration purpose, each regulation consists of two `FlowControlRuleMethods`, one `FlowControlPolicy` that contains these rules, and one `FlowControlMetaPolicy`. The first flow control rule is an application-specific subclass of `DenyingFlowControlRuleMethod` and blocks access to the computer network `cn-1`. The second flow control rule corresponds to a subclass of `AllowingFlowControlRuleMethod` and modifies the blocking of the first rule by allowing smaller parts of the network to be accessed again.

Each flow control rule covers the same regulated content, rule data provider, and sender. The topic of the regulated content is `t-1` and corresponds to neo-Nazi material. The rule data provider is `bka-1` and corresponds to the BKA which is in charge of creating flow control rules and sending them to the access provider. The sender of each rule and thus the requester of the regulated content is modeled as a whole computer network rather than a single computer. It is identified as `cn-2` and associated with the network address `195.145.0.0/16`. The specific definitions of the computer network `cn-2` and its details are modeled in Fig. 13(a). This network is located in Germany and managed by the German Telecom. The code of conduct `coc-1` of the German Telecom [21] is applied for all three flow control policies. Similarly, all three policies are authorized by the same `FlowRegulationNorm` `stgb86-1` which corresponds to §86 of the German Criminal Code that prohibits the distribution of neo-Nazi material.



(a) Representation of the German client computer network

(b) Representation of the neo-Nazi computer network

Fig. 13. General definitions of the scenario networks.

For reasons of brevity, the following depictions of a flow control regulation cover only the main aspects of a regulation. The basic structure for the sender of a flow control rule

is defined in Fig. 13(a) and the same for all rules. The definition of the regulated computer network `cn-1` is depicted in Fig. 13(b). This definition is also used in several flow control policies. All three meta-policies define a global conflict solution algorithm and a global non-applicability algorithm as these algorithms are a mandatory part of the conflict resolution process. In contrast, the three flow control policies do neither define a local conflict solution algorithm nor a local non-applicability algorithm. Local conflict solution algorithms are only evaluated if a policy contains two or more contradicting rules. Since the flow control rules used in the following examples do not provoke any conflicts, the algorithm is omitted for simplicity reasons. On the other hand, local non-applicability algorithms are only evaluated if an enforcing system cannot implement a particular flow control rule. The flow control rules used in the examples are basic blocking and allowing rules, respectively. The following subsections provide three examples of flow control rules, flow control policies, and flow control meta-policies to be enforced on a router, a name server, and a proxy server. A detailed workflow of creating and implementing these regulations is provided on InFO's website, which is referenced in the conclusion.

5.1. *Applying the Router Ontology*

An example regulation using the Router Ontology is depicted in Fig. 14. It shows two flow control rules, one flow control policy, and one flow control meta-policy. All regulations are enforced by the router `ro-1` which is operated by the German Telecom (not depicted in every figure for reasons of brevity). The first rule `r-1` is depicted in Fig. 14(a) and states that any computer from the network `cn-2` shall be prevented from establishing a connection to the Stormfront network `cn-1`. The computer network `cn-2` is located in Germany and managed by the German Telecom `tcom-1`. The intention of rule `r-1` is to prevent German users of the network `cn-2` from accessing neo-Nazi material hosted within the Stormfront network `cn-1`. However, the network also includes servers which do not provide neo-Nazi material such as the name server `sf-ns-1`. Thus, the flow control rule `r-2` depicted in Fig. 14(b) allows to access this name server. The rule `r-2` is also managed by the German Telecom `tcom-1`, which is not shown in the figure for reasons of brevity. Furthermore, `r-2` also shares the same rule data provider, regulated topic, and sender network as rule `r-1`.

The flow control policy `p-1` depicted in Fig. 14(c) associates the rules `r-1` and `r-2` with their legal authorization and their organizational motivation. The legal authorization is §86 of the German Criminal Code `stgb86-1`. The organizational motivation corresponds to the German Telecom's code of conduct `coc-1`. Since the policy does not define any local conflict solution algorithm, all conflicts between conflicting rules are resolved by the policy's meta-policy `mp-1` depicted in Fig. 14(d). It contains the flow control policy `p-1`. Both rules `r-1` and `r-2` cover the same IP address region `ipr-1` which is associated with the name server `sf-ns-1`. Thus, both rules can in general be applied to this server. In order to decide which of the two rules are to be used, the enforcing router `ro-1` must apply its meta-policy's rule priority algorithm `rpa-1`. This algorithm requires an explicit ordering of flow control rules with the property `follows`. As depicted in Fig. 14(c), the rule `r-2`

has a higher priority than r-1 in order to allow access to the name server sf-ns-1. Thus, the router ro-1 applies the rule r-2 to the name server sf-ns-1 and the rule r-1 to all other servers of the computer network cn-1. This results in the refusal of any communication attempts to any server in the network 174.121.229.0/24 except for the name server with the IP address 174.121.229.147.

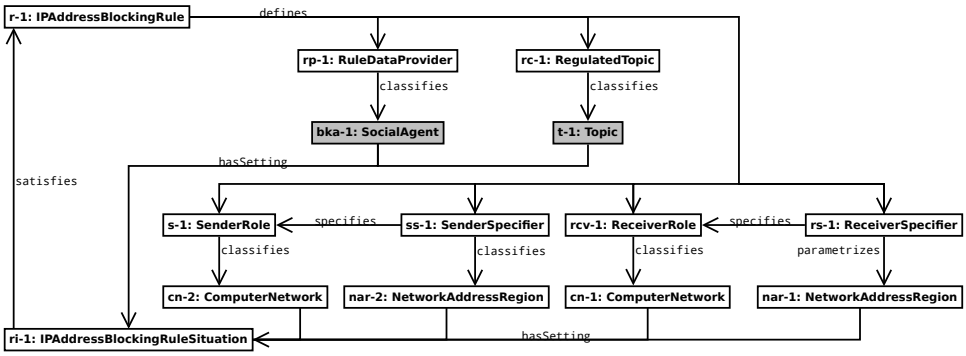
5.2. *Applying the Name Server Ontology*

An example regulation using the Name Server Ontology is depicted in Fig. 15. The regulation is enforced by the name server ns-1 operated by the German Telecom and consists of two flow control rules, one flow control policy, and one flow control meta-policy. The first rule r-4 depicted in Fig. 15(a) states that any German client connected to the network cn-2 shall be prevented from establishing a connection to the network cn-1. The intention of this rule is the same as the intention of rule r-1. Again, access to the name server sf-ns-1 is explicitly allowed by the second flow control rule r-5 which is depicted in Fig. 15(b).

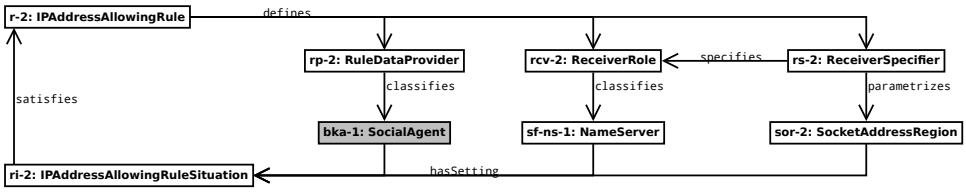
The flow control policy p-2 of the rules r-4 and r-5 is depicted in Fig. 15(c). Similar to the flow control policy p-1, p-2 also does not define any local conflict solution algorithm or a local non-applicability algorithm itself. Therefore, non-applicable rules and conflicts between contradicting rules are handled by the policy's meta-policy mp-2 depicted in Fig. 15(d). Since both the rule r-4 and r-5 cover the domain name dnr-1 of the name server sf-ns-1, they can generally be applied for regulating the access to this server. Again, the enforcing name server ns-1 must apply its meta-policy's rule priority algorithms. In this case, the meta-policy defines two different algorithms which are rpa-2 and rpa-3. As depicted in Fig. 15(d), rpa-2 has a higher priority than rpa-3 and must be applied first. The algorithm rpa-2 states that longer domain names shall be preferred to shorter ones. In this case, the domain name iserver.stormfront.org of the name server sf-ns-1 is longer that the domain name stormfront.org of the whole computer network cn-1. The name server ns-1 thus applies the rule r-4 to the name server sf-ns-1 and the rule r-5 to all other servers of the same domain. This results in the same effect as the flow control enforced by the router ro-1. Communication attempts to the name server with the domain iserver.stormfront.org are allowed whereas access to all other servers of the domain stormfront.org are blocked.

5.3. *Applying the Application-Level Proxy Ontology*

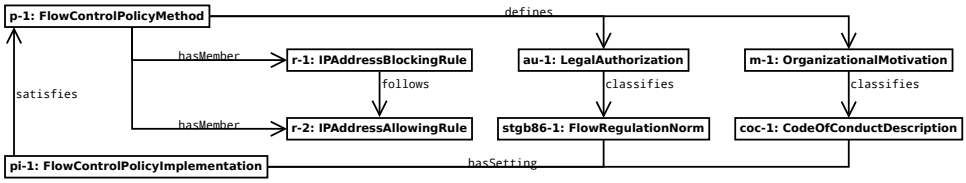
An example regulation applying the Application-Level Proxy Ontology is depicted in Fig. 16 and enforced by the proxy server pr-1. Again, the regulation consists of two flow control rules, one flow control policy, and one flow control meta-policy. The first rule r-7 is depicted in Fig. 16(a). It states that a German user of the network cn-2 shall be prevented from accessing the neo-Nazi web forum wst-1. Although the forum wst-1 may contain neo-Nazi material, the contact form wp-2 available at http://stormfront.org/forum/sendmessage.php does not. Therefore, the r-8 depicted in Fig. 16(b) allows accessing this particular web page.



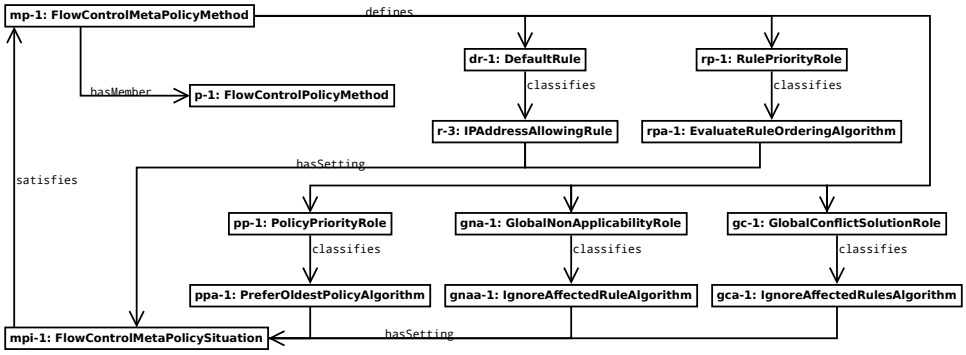
(a) First example flow control rule of the Router Ontology



(b) Second example flow control rule of the Router Ontology



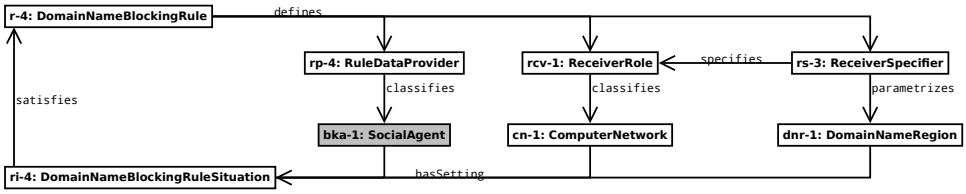
(c) Example flow control policy of the Router Ontology



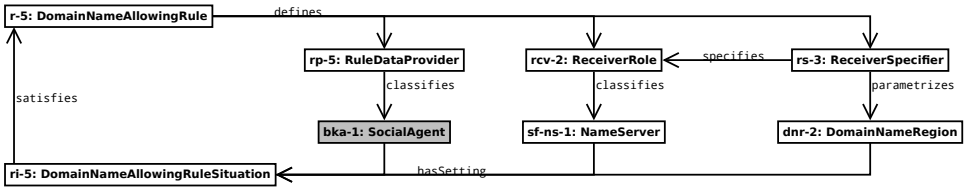
(d) Example flow control meta-policy of the Router Ontology

Fig. 14. Example usage of the Router Ontology

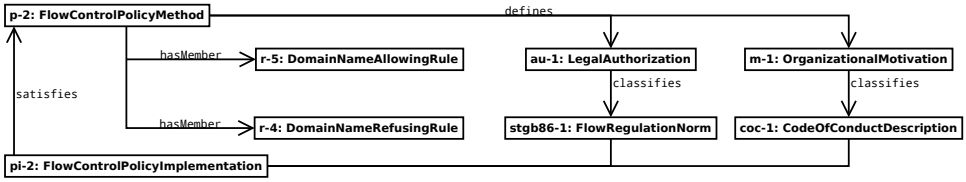
The flow control policy `p-3` combines the two rules `r-7` and `r-8` and is depicted in Fig. 16(c). Again, `p-3` does neither define a local conflict solution algorithm nor a local



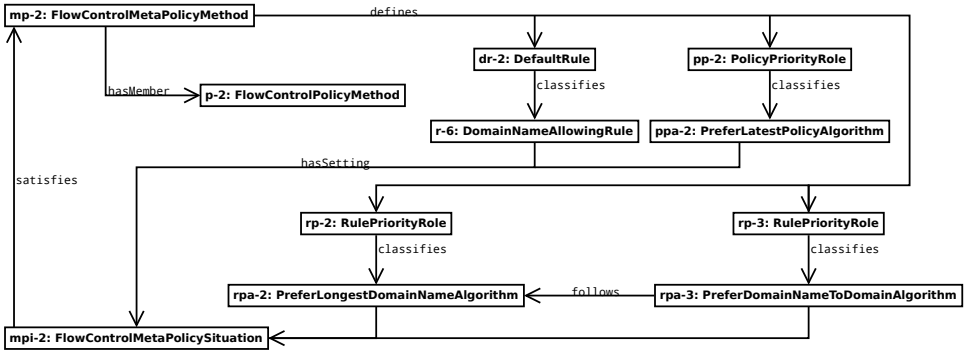
(a) First example flow control rule of the Name Server Ontology



(b) Second example flow control rule of the Name Server Ontology



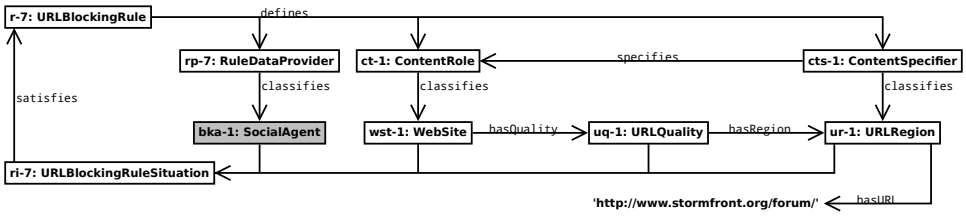
(c) Example flow control policy of the Name Server Ontology



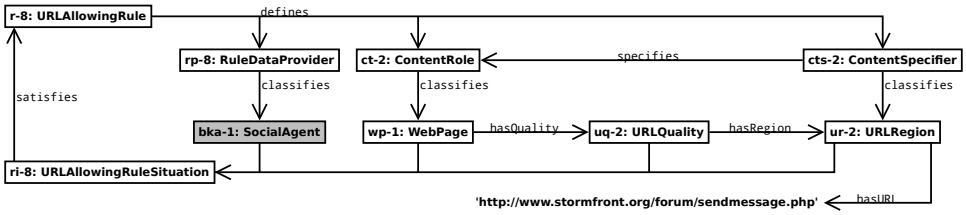
(d) Example flow control meta-policy of the Name Server Ontology

Fig. 15. Example usage of the Name Server Ontology

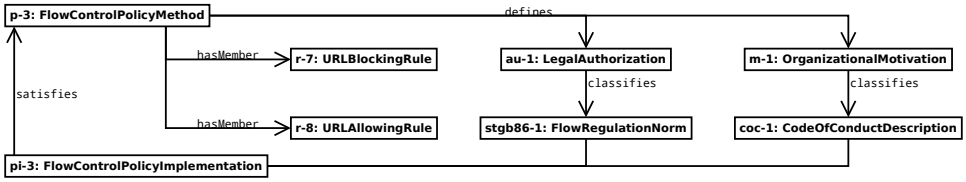
non-applicability algorithm itself and relies on its corresponding flow control meta-policy `mp-3` for resolving conflicts between contradicting rules and handling non-applicable rules. The flow control meta-policy `mp-3` is depicted in Fig. 15(d) and defines the algorithm `rpa-4` which is an instance of the class `PreferWebPageToWebSiteAlgorithm`. This algorithm states that rules associated with single web pages shall be preferred to rules with whole websites. The web forum `ws-1` corresponds to a website while the con-



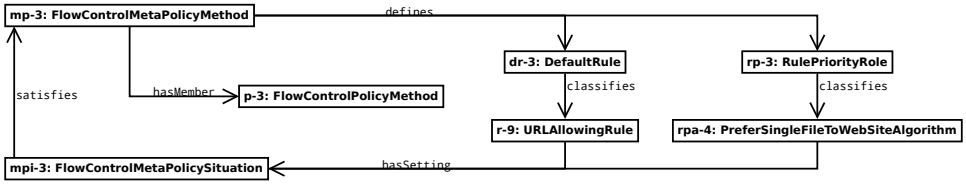
(a) First example flow control rule of the Application-Level Proxy Ontology



(b) Second example flow control rule of the Application-Level Proxy Ontology



(c) Example flow control policy of the Application-Level Proxy Ontology



(d) Example flow control meta-policy of the Application-Level Proxy Ontology

Fig. 16. Example usage of the Application-Level Proxy Ontology

tact form `wp-2` is a single web page. In applying this algorithm, the proxy server `pr-1` uses the rule `r-8` for allowing access to the web page `wp-2` while blocking access to all other web pages of the web forum `wst-1`.

6. Prototypical Implementation of the InFO Pattern System

The pattern system InFO and its three domain-specific extensions for routers, proxy servers, and name servers have been implemented on three prototypical systems. Each system consists of two different modules which are a *preparation module* and a *regulation module*. The preparation module resolves all possible conflicts of InFO policies and transforms the remaining flow control rules to a simpler data structure which can directly be used on

```
Internet Protocol Version 4
|--Source: 195.145.200.111 (195.145.200.111)
|--Destination: 195.145.23.155 (195.145.23.155)
|--Protocol: ICMP (1)
|--Options: (28 bytes)
+--Internet Control Message Protocol
   |--Type: 3 (Destination unreachable)
   |--Code: 13 (Communication administratively filtered)
   +--Internet Protocol Version 4
      |--Source: 195.145.23.155 (195.145.23.155)
      |--Destination: 174.121.229.156 (174.121.229.156)
      |--Protocol: TCP (6)
      +--Transmission Control Protocol
         |--Source port: 32517 (32517)
         +--Destination port: 80 (80)
```

Listing 1. Example blocking result of a router

the specific enforcing system. The common preparation module is implemented in Java for all three systems and uses the Jena triple store[f]. The regulation module performs the actual regulation by operating on the data structure created by the preparation module. Its implementation details depend on the corresponding system. The source code of the implemented systems is publicly available and can be downloaded from InFO's website, which is referenced in the conclusion.

6.1. *Example Router Implementation*

The Router Ontology has been implemented as a set of routers, which are configured via a dedicated administration node. This node serves as the preparation module and transforms all flow control rules into a format which can directly be used by the Linux firewall software iptables[g]. The administration node sends the iptables rules to all connected routers using a specific protocol. The iptables software runs on all routers and is directly used as regulation module. Thus, an additional implementation for the regulation module is not required. If a user wants to access a prohibited IP address, she receives an Internet Control Message Protocol (ICMP) [44] message that informs about the blocking. The ICMP is specifically designed for exchanging information messages and error messages between IP-based communication nodes. The ICMP message is encapsulated in an IP message and send back to the original requester. The sender of this message is the router implementing the flow control.

Listing 1 shows an ICMP message after a sender with the IP address `195.145.23.155` wanted to access a web server with the IP address `174.121.229.156`. The server is part of the computer network `174.121.229.0/24`, which is to be blocked according to the example flow control rule depicted in Fig. 14(a). The message was captured with

[f] `http://jena.apache.org` (last accessed: 03/11/14)
[g] `http://www.netfilter.org/projects/iptables/` (last accessed: 03/11/14)

```
1 < HTTP/1.1 451 Unavailable For Legal Reasons
2 < Content-Length: 362
3 < Content-Type: text/html; charset=iso-8859-1
4 <
5 <html>
6 <head><title>Unavailable For Legal Reasons</title></head>
7 <body>
8 <h1>Unavailable For Legal Reasons</h1>
9 <p>The web page you are trying to access is not accessible due to legal
10 reasons. For more information about the regulation see
11 <a href="http://icp.it-risk.iwvi.uni-koblenz.de/policies/proxyPolicy01.owl">the
12 regulation details</a>.</p>
13 </body>
14 </html>
```

Listing 2. Example blocking result of a proxy server

the packet analyzing software Wireshark[h] and slightly modified for illustration. It contains the header of the original request (lines 9 to 15). As depicted in Fig. 12, the node with the IP address `195.145.23.155` is located in Germany and `174.121.229.156` corresponds to the server hosting the website `http://stormfront.org/forum/`. The lines 12 and 15 further describe this server as a web server since such a server typically uses the port number `80` and the transport layer protocol TCP. The ICMP message shown in lines 6 to 15 contains the IP header of the original request and denotes that this request was blocked. Line 7 states that the web server with the IP address `174.121.229.156` was unreachable due to an administrative filtering as explained in line 8. The router's IP address is `195.145.200.111` (line 2). The options of the IP header (line 5) provide an encoded hyperlink with further background information of the regulation such as `http://icp.it-risk.iwvi.uni-koblenz.de/policies/ipPolicy01.owl` including its organizational and legal background. The detailed process of embedding a URL is described on InFO's website, which is referenced in the conclusion.

6.2. *Example Proxy Server Implementation*

The Application-level Proxy Ontology has been implemented as a prototypical proxy server, which is available at `141.26.83.115`. The preparation module of this proxy server stores all flow control rules in a relational database. If the regulation module receives a user request for a particular URL, it looks up the URL in the database and performs a corresponding regulation. The regulation module is implemented in Java and uses standard Java libraries.

An example request for `stormfront.org` using the Unix tool curl looks like: `curl -v http:// www.stormfront.org/forum/index.php`. This command requests the web page `http://stormfront.org/forum/index.php`, which is part of the web forum `http://stormfront.org/forum/`. The web forum is to be

[h] `http://www.wireshark.org/` (last accessed: 03/11/14)

```
1 ; <<>> DiG 9.8.1-P1 <<>> @141.26.83.113 +tcp -p 4053 stormfront.org
2 ; (1 server found)
3 ;; global options: +cmd
4 ;; Got answer:
5 ;; ->>HEADER<<- opcode: QUERY, status: REFUSED, id: 44811
6 ;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 9
7 ;; QUESTION SECTION:
8 ;stormfront.org. IN A
9 ;; ADDITIONAL SECTION:
10 stormfront.org. 3600 IN TXT "A:ID:0:http://...uni-koblenz.de/p.../dnsPolicy01.owl#r-7"
11 stormfront.org. 3600 IN TXT "A:PI:0:http://...uni-koblenz.de/p../dnsPolicy01.owl#p-3"
12 stormfront.org. 3600 IN TXT "A:PH:0:http://...uni-koblenz.de/p.../TelekomCoC.owl#coc-1"
13 stormfront.org. 3600 IN TXT "A:LW:0:http://...uni-koblenz.de/p.../StGB.owl#stgb86-1"
14 stormfront.org. 3600 IN TXT "A:DP:0:http://...uni-koblenz.de/p.../dnsPolicy01.owl#bka-1"
15 stormfront.org. 3600 IN TXT "A:TO:0:neo-Nazi material"
```

Listing 3. Example blocking result of a name server

blocked according to the example flow control rule depicted in Fig. 16(a). The response of the proxy server is depicted in Listing 2. The server returns the status code 451 (line 1) which indicates that the access to the web page was denied due to legal reasons [45]. Along with the status code, the proxy server also returns a short web page containing further information of the blocking reason (lines 5 to 14). It contains a hyperlink to the policy that initiated the flow control namely `http://icp.it-risk.iwvi.uni-koblenz.de/policies/proxyPolicy01.owl`.

6.3. *Example Name Server Implementation*

The Name Server Ontology has been implemented as a prototypical, modified name server, which is available at `141.26.83.113`. The preparation module of this name server requires minimal modifications of the DNS's record-based data structure. The data structure is downwards compatible with already existing name server implementations such as BIND[i]. The regulation module operates on this data structure. Its implementation is based on the Java name server EagleDNS[j] An example domain name request for the domain `stormfront.org` using the Unix tool dig looks like: `dig @141.26.83.113 +tcp -p 4053 stormfront.org`. As the example flow control rule depicted in Fig. 15(a) states, access to this domain name is to be blocked by the name server. The name server's response is depicted in Listing 3. The server `REFUSED` answering the request (line 5). Additional background information about the flow control is returned as several `TXT` records as shown in lines 10 to 15. The string `"A"` used in all `TXT` records states that these records refer to the requested IPv4 address record. `"ID"` in line 11 indicates that the `TXT` record contains the URI of the corresponding flow control rule. `"0"` corresponds to a local identifier of this URI and is used for grouping all `TXT` records which are associated with the

[i] `http://www.isc.org/downloads/bind/` (last accessed: 03/11/14)
[j] `http://www.unlogic.se/projects/eagledns` (last accessed: 03/11/14)

same rule. Since the `TXT` records shown in lines 10 to 15 all share the same local identifier, they are all based on the same flow control rule shown in line 11. If there is more than one rule which covers the same domain name, this number is used for distinguishing between them. Similarly, the string `"PI"` in line 10 denotes that the `TXT` record contains the URI of the corresponding flow control policy. The name server's response also contains further information about the flow control rule including its organizational background (line 12), its legal background (line 13), its rule provider (line 14), and its regulated topic (line 15). While the topic is directly embedded into the domain name response, further information about the other regulation details can be obtained by dereferencing the URI provided in the corresponding `TXT` records.

7. Related Work

Policy-based regulations of information processing can generally be distinguished between access control, flow control, and usage control. Classical access control focuses on regulating access to information at the content provider's site (i. e., the server) whereas usage control covers the regulation of information at the consumer's site (i. e., the client) [46, 47]. In contrast, flow control allows for regulating the flow of information between the provider and the consumer. This section extensively discusses policy languages for access control, usage control, flow control, and general purpose languages and compares them with InFO. Legal ontologies are not covered in this section since they can be integrated into InFO. The reader may refer to [48] for an overview of legal ontologies.

7.1. *Access Control Models*

The Access Management Ontology (AMO) [49] is an RDFS [50] ontology for describing access control rules for collaborative web-based document systems such as Wikis or Content Management Systems. In order to ease the creation of access rules and their integration in such systems, AMO's design is very simple and only allows for modeling permitted actions. Other types of rules such as prohibitions are not supported. Actions, which are not explicitly allowed by AMO, are considered forbidden. Common Policy [51] is an XML-based language for describing access rules for personal data. Similar to AMO, the language model of Common Policy is rather lightweight and only allows for defining permitted actions. Furthermore, Common Policy is designed to be used only in combination with additional application-level protocols such as FTP or HTTP. These protocols must cover the authentication of the requesting user and the transmission of the requested data. WebAccessControl[k] is another lightweight RDFS ontology for describing access control rules for web resources. It is designed for decentralized systems in which the web resources and the users can be managed by different parties. All users are identified by their WebID[l] which serves as a globally unique identifier. User authentication is done using the WebID authentication protocol. Similar to AMO and CommonPolicy, WebAccessControl has a simple

[k] `http://www.w3.org/wiki/WebAccessControl`, accessed: 10/06/14
[l] `http://www.w3.org/wiki/WebID`, accessed: 10/06/14

design which only supports permitted actions. The Enterprise Privacy Authorization Language (EPAL) [5] and the eXtensible Access Control Markup Language (XACML) [4] are XML-based access control languages which allow for creating much more expressive policies than AMO, Common Policy, or WebAccessControl. Both languages are designed to be used within closed network environments such as intranets of large corporations. While EPAL merely focuses on regulating access to personal data, XACML does not have a predefined use case and can be used for regulating the access to arbitrary data. Specific use cases are implemented by creating corresponding XACML profiles. A profile for regulating access to personal data is given in [52]. According to [53], XACML is much more expressive than EPAL and can replace it in many applications.

In summary, classical access control regulates access to data within a closed environment [46, 47]. XACML and EPAL can be considered as classical access control systems since they require a centrally controlled enforcing infrastructure. On the other hand, AMO, Common Policy, and WebAccessControl focus on regulating access to pieces of information in a rather open environment such as the web. In all access control systems, content providers regulate the access to their content. However, such regulations cannot be used for regulating the communication flow between an arbitrary server and arbitrary client on the Internet.

7.2. *Usage Control Models*

Rights Expression Languages (REL) allow for defining usage control policies for digital objects. RELs often define only an abstract policy model which is accompanied by an additional Rights Data Dictionary (RDD). The REL covers the basic syntax for creating policies while the RDD is used as a vocabulary to create specific policies. ccREL [54] is a lightweight RDFS ontology primarily designed for describing Creative Commons[m] licenses. Such licenses describe actions that may, must, or must not be applied on the digital good. In order to be easy to use, there are six pre-defined licenses which can be applied to arbitrary goods. Although ccREL can generally be extended with additional terms, using such terms may lead to licenses which do not correspond to Creative Commons licenses. The Linked Data Rights ontology (LDR) [55, 56] is a lightweight OWL ontology which supports usage control licenses for linked data resources. Although it defines a few terms itself, it is mainly designed to be extended with additional terms for particular use cases. The Metadata Encoding and Transmission Standard (METS) [57] is a general language model for describing different types of metadata for digital resources. METS itself only defines a basic XML language structure which must be extended with additional vocabularies in order annotate the digital resource. METSRights[n] is an example vocabulary which provides a very basic REL. It only allows for defining which parties are allowed to perform which actions on a digital resource. It does not provide any means for defining prohibitions. More complex usage control languages are MPEG-21 REL [58] and the Open

[m] `http://creativecommons.org/`, accessed: 10/06/14
[n] `http://www.loc.gov/standards/rights/METSRights.xsd`, last accessed: 03/11/14

Digital Rights Language (ODRL) [37, 38]. MPEG-21 REL is part of MPEG-21 which is a language framework similar to METS for annotating different types of metadata to digital resources. Both MPEG-21 REL and ODRL can be used for the same applications and allow for creating almost arbitrary usage control policies. The two languages are XML-based and define their own REL and RDD. However, the default RDD is not mandatory when creating specific policies. Instead, the creation of user-defined RDDs are also possible. In ODRL, user-defined RDDs are called profiles. An example profile is RightsML [59] which is designed for managing usage rights in the news industry.

RELs allow for describing which users are permitted to perform which actions on which digital resources. Contrary to flow control languages, these descriptions are rather abstract and must be further interpreted in order to enforce them. The more abstract a particular policy is, the more interpretations and implementations of the same policy are possible. For example, ODRL's RDD defines the action *anonymize* as the "act of anonymising the asset" [38]. This action cannot be directly enforced by a system since it usually does not have a precise understanding of what and how much data shall actually be removed from the original asset. Since RELs usually do not provide a precise description of their policies, different interpretations of the same policy may all be considered valid.

7.3. *Flow Control Models*

The XML-based firewall metamodel proposed by Cuppens et al. [8], the UML-based DEN-ng [6], and the OWL-based Ontology-Based Policy Translator (OPoT) [7] are models for describing flow control. They aim at easing the management of a closed network environment supervised by a single organization or institution by mapping high-level organizational security policies to different network systems such as routers. While both the firewall metamodel and DEN-ng merely focus on low-level routers and do not directly support communication end points such as web servers, OPoT also covers different nodes of a communication path including the end systems. The firewall model of Cuppens et al. only supports permitting rules. It does not provide any means for defining explicit prohibited communication flows. Any communication that is not explicitly allowed is considered forbidden. OPoT uses a set of twelve predefined basic policies which each cover a specific use case. A basic policy can be considered a template for implementing a specific organizational security policy. In order to actually enforce such an organizational policy, a corresponding basic policy has to be chosen and mapped to the current network environment. This mapping corresponds to filling in the basic policy with specific IP addresses and other implementation details.

In summary, flow control languages focus on regulating the flow of communication within a closed network environment which is centrally administrated by a single entity. The existing languages are designed to allow a direct enforcement of their policies by network nodes without requiring any further interpretation. This is achieved by describing the actions which must be performed by the nodes including all necessary parameters such as IP addresses, port numbers, and communication protocols.

7.4. *General Purpose Models*

General purpose languages do not focus on one particular type of information regulation but rather follow a more general approach in order to cover several scenarios such as access control or flow control. KAoS [60], Rei [61], and Ponder [36] are examples for such languages. Since these languages allow for different types of control policies, they cannot be clearly categorized as access control, usage control, or flow control.

KAoS is based on OWL and allows for creating policies which describe what systems a user can access within a closed management environment such as an organization. An example for such a policy is granting a user access to a specific server. However, KAoS only allows for regulating access to the server or its provided services and cannot further distinguish between the data hosted by them. More specifically, KAoS does not provide any means for directly regulating the content hosted by a server. A KAoS policy can generally be enforced by any application-level communication system including content providers, content consumers, and application-level proxy servers. Contrary to KAoS, both Rei and Ponder allow for defining document-centric policies. This allows for creating more precise access control policies. Rei is based on OWL and merely considers reasoning about policies and is not designed for enforcing them [62]. Ponder uses its own low-level object-oriented language syntax which is not compatible with W3C standards such as XML or RDF. The language is able to define policies which can be enforced on the content providers' server, the end users' clients, as well as on intermediary communication nodes. However, due to its low-level descriptive language, Ponder is not able to cover different levels of abstraction on the same regulation [62] such as organizational or legal background information.

The applicability of a general purpose language as well as the implementation and enforcement of its policies heavily depends on the language's design. While KAoS and Rei focus on a more abstract view of a policy, Ponder merely covers its technical implementation details.

7.5. *Detailed Comparison and Discussion*

Table 2 assesses the related work according to the functional and non-functional requirements introduced in Section 3 and compares it with InFO. Most of the reviewed policy languages focus on a particular application and thus do not fulfill all requirements stated in Section 3. In the following, the related work is discussed along the functional requirements **(1)** to **(8)** as well as the non-functional requirements **(9)** and **(10)**.

Access control as introduced in Section 7.1 is only possible at the application layer since it requires the user-based management of the content being accessed. Thus, access control languages such as AMO, Common Policy, WebAccessControl, EPAL, and XACML exclude routers as possible enforcing systems since they operate on lower layers of the OSI model such as the network layer and possibly the transport layer as well. EPAL and XACML are designed to be used separately from the server providing the content while AMO, Common Policy, and WebAccessControl require a more closely integration with the server. This allows EPAL and XACML policies to be enforced by proxy servers **(1c)**, while policies created with AMO, Common Policy, or WebAccessControl can only be enforced

Table 2. Comparison of the capabilities of different policy languages with the requirements introduced in Section 3. Rows correspond to policy languages and columns to requirements. Requirements **(1)** to **(8)** are functional, while **(9)** and **(10)** are non-functional. The letter y corresponds to a complete fulfillment of the requirement, l corresponds to a partial fulfillment, and n corresponds to no fulfillment of the requirement.

	Support for routers as enforcing nodes **(1a)**	Support for name servers as enforcing nodes **(1b)**	Support for proxy servers as enforcing nodes **(1c)**	Operationalization of policies **(2)**	Rules with allowing modality **(3a)**	Rules with denying modality **(3b)**	Rule conflict solution for the same policy **(4a)**	Rule conflict solution for different policies **(4b)**	Regulation enforcer, provider, legislator **(5a)**–**(5c)**	Content identification **(6a)**	Content classification **(6b)**	Access location **(7)**	Organizational background **(8a)**	Legal background **(8b)**	Modularity **(9)**	Extensibility **(10)**
Access Control																
AMO	n	n	n	n	y	n	n	n	n	y	n	n	n	n	n	y
CommonPolicy	n	n	n	n	y	n	n	n	n	n	y	n	n	n	n	y
EPAL	n	n	y	l	y	y	l	n	l	n	y	y	n	n	n	y
WebAccessControl	n	n	n	y	y	n	n	n	l	y	n	n	n	n	n	y
XACML	n	n	y	l	y	y	y	y	l	y	y	y	n	n	n	y
Flow Control																
Cuppens et al.	y	n	n	y	y	n	n	n	n	n	n	y	y	n	n	n
DEN-ng	y	n	n	y	y	y	l	l	n	n	n	y	n	n	n	y
OPoT	y	n	n	y	y	y	l	l	n	n	n	y	y	n	n	n
Usage Control																
ccREL	n	n	n	n	y	y	n	n	n	y	n	n	n	l	n	l
LDR	n	n	n	n	y	y	n	n	n	y	y	n	n	l	n	y
ODRL	n	n	n	n	y	y	n	l	l	y	n	y	n	n	l	l
METSRights	n	n	y	n	y	n	n	n	l	y	l	n	n	n	n	l
MPEG-21 REL	n	n	n	n	y	n	n	n	l	y	n	n	n	n	l	l
General purpose																
KAoS	n	n	y	n	y	y	y	n	l	n	n	y	y	n	y	y
Rei	n	n	y	n	y	y	l	n	l	y	y	y	n	n	y	y
Ponder	y	n	y	y	y	y	y	n	n	y	y	y	n	n	n	y
InFO	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y

by the content providing server. Although the flow control languages DEN-ng, OPoT, and the firewall metamodel focus on managing low-level enforcing communication nodes such as routers **(1a)**, they are not designed to define policies enforced by proxy servers or name servers. Usage control policies also require an enforcement at the application layer and

are generally enforced at the user's site. ccREL, LDR, ODRL and MPEG-21 REL thus do not fulfill requirements **(1a)**–**(1c)**. However, METS is designed to be used within closed library environments making METSRights suitable to be enforced at the library's proxy server **(1c)**. KAoS and Rei focus on rather abstract behavioral policies which are also designed to be enforced by application-layer systems. This makes it possible to enforce their policies by proxy servers. On the other hand, Ponder allows for defining policies which can be enforced by almost arbitrary communication nodes including end user systems, content-providing servers, or intermediary communication nodes such as routers. However, neither Ponder nor any other of the considered policy languages support name servers as enforcing nodes **(1b)**. Name servers are not part of the communication path between a content provider and a content consumer. Instead, they only provide a means for establishing this communication path, which is not covered by any of the languages depicted in Table 1. InFO, on the other hand, allows for using different enforcing systems including routers, name servers, and proxy servers. Each enforcing system is supported by a specific domain ontology such as the Router Ontology, the Name Server Ontology, and the Application-Level Proxy Ontology.

Many of the examined policy languages define rather abstract rules whose enforcement cannot be directly mapped onto the enforcing systems' capabilities. Instead, the enforcement requires additional parameters and a further interpretation of how to interpret the policy's actual meaning. These parameters are sometimes not directly included in the policy **(2)** and must therefore be added through a different process. Although policies created with AMO or Common Policy contain a reference to those users who are allowed to access a specific piece of information, they do not define how the users shall be authenticated. EPAL and XACML do provide such a description but do not explicitly define the rest of the enforcement procedure. WebAccessControl requires user identification via the WebID authentication process. Usage control only describes on an abstract level what a user may do with a digital resource. However, usage control languages do not define how the permitted or prohibited actions shall actually be regulated. For example, it is unclear how a permission to print a text document is to be enforced. The evaluated usage control policies thus do not fulfill requirement **(2)**. Since the considered general purpose languages Rei and KAoS also define rather abstract policies and not their specific enforcement, they also do not fulfill requirement **(2)**. However, Ponder's low-level language can directly be used for enforcing mechanisms. On the other hand, flow control languages are specifically designed for a direct enforcement of policies. Each created policy already contains enough information to be enforced without requiring any additional interpretation or parameters. Similarly, InFO is also designed for enforcing particular policies. Support for a precise description of all enforcement parameters are provided by the domain extensions of InFO such as the Router Ontology, the Name Server Ontology, and the Application-Level Proxy Ontology.

All policy languages shown in Table 1 support allowing rules and thus fulfill requirement **(3a)**. In order to ease the creation of specific policies, some languages such as AMO, Common Policy, WebAccessControl, and the firewall metamodel prohibit denying rules **(3b)** and focus on allowing rules only. In doing so, these languages completely avoid potential conflicts between two or more contradicting rules. They thus do not provide any

means for resolving such conflicts (requirements **(4a)** and **(4b)**). InFO allows for creating both allowing and denying rules as well as specific types of denying rules. The different types of rules are provided by the Flow Control Rule Pattern as well as the Redirecting Flow Control Rule Pattern and the Replacing Flow Control Rule Pattern described in Section 4.3. The usage control policy languages ccREL and LDR assume that there is only one policy for each regulated good which is created by its owner. Thus, the languages do not provide any means for resolving conflicts between contradicting rules. EPAL and Rei only provide the ordering of rules as a mechanism for resolving conflicts between rules of the same policy **(4a)**. Since both languages support only one active policy, they do not provide any means for resolving conflicts between rules of different policies **(4b)**. DEN-ng also uses the ordering of rules for resolving conflicts between them. Additionally, policies can also be ordered to resolve conflicts between rules of different policies. OPoT is only able to detect conflicting rules of one or more policies and shows them to the policy creator. The actual conflict solution must be performed manually by the policy creator. ODRL assumes that a single policy does not contain any conflicting rules. Since each policy is created by a single party, the party must pay attention to the created policies. However, ODRL allows for resolving conflicts between rules of different policies. This is done by either preferring allowing or denying rules. XACML and Ponder allow for defining specific conflict solution algorithms which provide a much greater flexibility than a simple ordering of rules. These algorithms can be used for resolving conflicts between rules of one or more policies. KAoS also provides algorithms for resolving conflicts. Since KAoS only supports one active policy, these algorithms can only be used for contradicting rules of one policy. The conflict solution mechanisms of InFO are inspired by XACML. Similar to XACML and Ponder, InFO allows for resolving conflicts between contradicting rules of one or multiple policies based on predefined or user-defined algorithms. The Flow Control Meta-Policy Pattern described in Section 4.5 is especially designed for expressing such algorithms and thus for resolving conflicts as well. The pattern also splits the whole conflict solution process of XACML and Ponder into four different steps and assigns a particular algorithm to each step. This allows for a greater flexibility when some algorithms can be reused for different enforcing systems while others must be replaced with more specific algorithms.

Most of the languages do not distinguish between a policy's creator (i. e. the provider) and its enforcer. Languages like EPAL, XACML, ODRL, METSRights, MPEG-21 REL, KAoS, and Rei, which allow naming a policy's provider within the policy itself **(5b)**, do not allow for naming a separate enforcer **(5a)**. However, InFO explicitly requires such a distinction as outlined in the scenario of Section 2. Support for a regulation's provider is given as the rule data provider defined in the Flow Control Rule Pattern and a means for defining the regulation's enforcer is given in the Flow Control Policy Pattern described in Section 4.4. Most of the reviewed policy languages are not able to link a policy to its legal background **(8b)**. Consequently, they do not allow for specifying the legislator of the policy's legal background **(5c)**. Both ccREL and LDR allow policies to be linked to their jurisdiction. This jurisdiction defines the circumstances under which a policy is valid and may even add additional permissions or prohibitions. However, the jurisdiction only refers to a country's legislation and not to particular laws. Identifying the creators of

this legislation is neither supported by ccREL nor by LDR. InFO supports both relating a flow control to its legal background and the definition of a legislator as well. The legal background is described by the Flow Regulation Norm Pattern introduced in Section 4.6 and the legislator is part of the Legislation Pattern. By linking technical policies to their legal background, InFO allows for a better comparison between different policies of various enforcing systems.

The identification and classification of content **(6)** is only supported by those policy languages which are able to directly regulate the processing of particular information documents rather than whole systems or services only. Such languages include access control and usage control languages as well as most general purpose languages. Both AMO and WebAccessControl require the explicit identification of the document to be protected by using an URI **(6a)**. Content classification is neither supported by AMO nor by WebAccessControl. On the other hand, both Common Policy and EPAL only support classes of documents but do not allow for a more precise identification of the content **(6b)**. Data classification with Common Policy can be achieved by using the *sphere* constraint whereas EPAL provides *data categories*. XACML allows for regulating access based on either the contents' ID or its topic. Usage control languages are designed to control the consumption of digital resources. Thus, they all support a precise identification of the content to be controlled. METSRights also supports a simple classification of the content according to its licensing status such as *copyrighted* or *licensed*. However, an actual content description is not supported. Flow control languages only focus on regulating communication between complete systems and thus do not fulfill requirements **(6a)** and **(6b)**. Although InFO also focus on regulating flow control, it also considers the topic of the regulated content. The Flow Control Rule Pattern associates each particular flow regulation with such a topic.

The location of the user who wants to access a controlled resource can be implemented in different ways. EPAL, XACML, ODRL, KAoS, Rei, and Ponder support constraints regarding the applicability of their rules. These constraints also cover location constraints which directly implement requirement **(7)**. All evaluated flow control languages are able to regulate network communication using IP addresses. Since these addresses can be mapped to a geographical location, the flow control languages also fulfill requirement **(7)**. InFO also uses IP addresses to refer to a requesting user's country and thus supports requirement **(7)**.

Organizational background information corresponds to the enforcer's motivation to implement a specific policy. Although some of the considered policy languages provide a *purpose* constraint, this property does not correspond to an actual explanation of a policy's meaning and function. Instead, it only restricts the applicability of allowing and denying rules to a specific use case. The firewall metamodel and OPoT are designed for mapping high-level organizational security policies to their technical representation. Such a design also allows for linking policies created with one of these languages to their corresponding security policy. In doing so, the policies are enriched with a human-readable description and thus implement requirement **(8a)**. KAoS follows a different approach by directly embedding a human-readable description into a created policy using the property `hasDescription`. InFO allows for expressing a regulation enforcer's code of conduct with the Code of Conduct Pattern described in Section 4.6.

Regarding the non-functional requirements, it can be stated that none of the considered access control languages has a modular design and thus do not fulfill requirement **(9)**. Instead, their specification consists of a single document which cannot be further partitioned into different sections. However, the main entities defined by the languages can still be extended with additional terms such as new actions or new roles **(10)**. The examined flow control languages solely focus on network management and already provide a sufficient vocabulary for expressing corresponding policies. Due to their restricted use case and their straightforward design, they do not provide a modular structure or a broad extensibility. However, based on its open design on UML, DEN-ng is still able to be extended with additional language elements. Based on their lightweight design, neither ccREL nor LDR have a modular structure. Although they can both be extended with additional terms, using such terms in a ccREL policy may result in a policy which no longer corresponds to a Creative Commons license. ODRL and MPEG-21 REL define an REL and a separate RDD. Since the default RDD is not mandatory and can be replaced with a user-defined one, the separation between the REL and the RDD can be considered as limited modularity. However, the REL of both languages itself is not modular. The extendability of both ODRL and MPEG-21 REL is limited to defining new vocabulary terms for their corresponding RDD such as new actions or constraints. Defining new entities to their REL's model is not possible. METSRights does not define separate specifications for an REL and an RDD and thus cannot be considered modular. However, it is still possible to add new terms to the language's vocabulary. KAoS and Rei are based on OWL. The concepts of these languages are separated into different ontologies which each cover a specific aspect of them. For example, both languages define an ontology for describing actions and a separate ontology for policies. Since the languages are based on OWL, they also fulfill requirement **(10)** by supporting user-defined extensions. Although Ponder also supports the definition of new language entities such as new rule types, its proprietary representation format does not allow for a modular design **(9)**. InFO's modular design consists of several ontology design patterns **(9)**. Many of these patterns can be extended with new concepts such as introducing new rule types as a subclass of `FlowControlRuleMethod`. Furthermore, InFO is specifically designed to be extended with domain-specific ontologies that cover concepts relevant for particular use cases.

7.6. *Summary*

None of the considered policy languages can be used for solving the problem of flexible information flow control on the Internet. However, InFO reuses some of their concepts such as meta-policies and different conflict resolution algorithms. InFO's extendability is inspired by Ponder which allows for modeling arbitrary types of communication flow. XACML's flexible conflict solution algorithms are also adopted by InFO. Since InFO is a pattern system which covers a core ontology, ontological languages such as AMO, WebAccessControl, KAoS, and Rei may be aligned as domain specific extensions.

8. Limitations and Future Extensions

This section first discusses some limitations of InFO w.r.t. interpreting laws and legal documents and mapping them to technical regulation details. Subsequently, InFO is compared to Software Defined Networking (SDN) and similarities as well as differences between both concepts are discussed.

InFO provides a solution for a technical regulation of Internet communication without requiring any manual interaction. The regulation's organizational background and legal background are primarily used as its human-readable explanation. This background information is expressed using external ontologies which are integrated with InFO. However, even existing legal ontologies may not be able to completely replace every human intervention. As Brown and Greenberg [63] have demonstrated, not all legal cases are formally decidable and require manual interaction instead. Thus, InFO considers the mapping from an organizational background and/or a legal background to a technical regulation to be a manual process. In addition, some legal regulations define exceptions concerning the human user. For example, the German Criminal Code contains several norms related to computer crimes including §202c [64]. §202c prohibits the creation and distribution of software tools which can be used for conducting such crimes. However, §202c does not apply for security experts who use the software tools for assessing the security of a computer system [65]. The treatment of such exceptions generally requires human intervention such as it is done by courts [65]. A completely automatic assessment of a particular situation is not always possible. Even a security expert may violate §202c if she uses software tools to deliberately sabotage a computer system without having a proper authorization. Although InFO can easily be extended with additional roles representing the intervening parties, the actual interpretation of the exceptions would still require a manual intervention.

InFO's focus is the regulation of information flow in open networks such as the Internet. However, it can also be used for regulating information flow within closed networks or intranets which are centrally administrated by a single organization. Software Defined Networking (SDN) [66] defines a generic architecture for flexible and dynamic management of such networks. SDN generally distinguishes between the logical view of the network and its physical implementation. The former is provided by a central control node, called the control plane. It manages and configures the forwarding tables of all other network nodes such as routers and switches. These network nodes correspond to the physical part of the network and are called the forwarding plane. They carry out the actual packet forwarding using their forwarding tables. In SDN, the control plane configures the network nodes of the forwarding plane via a specific protocol. This protocol and the distinction between the control plane and the forwarding plane are the main components of the SDN architecture. Since SDN defines a generic architecture, a particular implementation of all three components is not provided. Instead, different protocols and even different routers and switches can be used. The only requirement is that all three components of the architecture are compatible with each other. Although SDN defines a generic architecture for managing networks, it primarily focuses on packet forwarding. Thus, the only types of communication nodes configured within SDN are routers and switches. Other network

nodes such as name servers and switches are not supported. In contrast, InFO provides a rich vocabulary for defining rules for regulating Internet communication. The regulations cannot only be implemented on routers and switches like with SDN, but also on name servers and proxy servers. Furthermore, InFO also attaches an organizational and/or legal foundation of a technical regulation. Thus, in summary InFO differs from SDN w. r. t. the following aspects: (1) InFO defines a language for specifying rules for regulating communication. It does not define any restrictions how to implement these rules on a particular network node. In addition, InFO also does not define any protocols for exchanging the rules between the nodes enforcing the regulation of the communication in a network. On the other hand, SDN defines a concrete architecture including a configuration protocol for all involved network nodes. (2) SDN requires a central control node for managing all regulations. Such a central node is not required with InFO. In fact, issuing of regulating rules is transparent to InFO. (3) InFO enriches technical specifications of regulation rules with human-readable descriptions. While the regulation specifications can directly be implemented by technical enforcing nodes, the human-readable description provides a reference to some organizational and/or legal background. In contrast, SDN is primarily concerned with technical implementation details. Regulations described with InFO can be implemented on routers, switches, name servers, and proxy servers. Furthermore, it is even possible to develop InFO-compatible search engines. On the other hand, SDN only supports routers and switches. Although InFO and SDN significantly differ in their general approach and their basic features, they can also be used together. More specifically, InFO can be a used as part of the SDN protocol for exchanging regulation information between the control plane and the forwarding plane. However, this would still require designing and implementing a complete protocol stack for both the control plane and the forwarding plane. A first approach towards this integration has been conducted by implementing policy-based regulation rules on routers which is described in Section 6.1.

9. Conclusion

This paper has presented the pattern system InFO (short for: Information Flow Ontology) as a flexible approach for information flow control on the Internet. InFO can be applied at different intermediate communication nodes such as routers, proxy servers, or name servers by providing a meta-language for existing as well as possible future regulation types. This is achieved by ontology-based policy descriptions abstracting from the existing solutions for information flow control on the Internet. InFO is based on the formal, pattern-based upper ontology DOLCE+DnS Ultralite (DUL). Each pattern solves a different aspect of information flow control. In addition, the patterns are designed to be applied together and to be extended with respect to further domain-specific requirements. Various examples of applying InFO and a prototypical implementation show the practical applicability of the approach. The detailed axiomatization of the patterns of InFO can be obtained from the ontology files provided in OWL. The ontology files of the individual patterns, the domain-specific extensions of InFO for name servers, routers, and application-level proxy servers,

the presented example policies as well as additional examples are all available at: `http://icp.it-risk.iwvi.uni-koblenz.de`.

In the future, one may define additional pattern systems for access control and usage control. These may be developed in combination with the already existing patterns of the InFO pattern system on information flow control on the Internet.

References

[1] Ravi Sandhu and Pierangela Samarati. Access control: Principles and practice. *IEEE COMMUN MAG*, 32(9):40–48, 09 1994.

[2] Yulia A. Timofeeva. Hate speech online: Restricted or protected? Comparison of regulations in the United States and Germany. *J. Transnatl. Law Policy*, 12:253–286, 2002.

[3] Ronald Deibert, John Palfrey, Rafal Rohozinski, and Jonathan Zittrain, editors. *Access Controlled: The Shaping of Power, Rights, and Rule in Cyberspace*. MIT Press, 2010.

[4] Tim Moses. eXtensible Access Control Markup Language (XACML) version 2.0. Oasis standard, 02 2005. `http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf` (last accessed: 03/11/14).

[5] Paul Ashley, Satoshi Hada, Günter Karjoth, Calvin Powers, and Matthias Schunter. Enterprise Privacy Authorization Language (EPAL 1.2), 10 2003. `http://www.w3.org/Submission/2003/SUBM-EPAL-20031110/` (last accessed: 03/11/14).

[6] John Strassner, José Neuman de Souza, Sven van der Meer, Steven Davy, Keara Barrett, Dave Raymer, and Srini Samudrala. The design of a new policy model to support ontology-driven reasoning for autonomic networking. *JNSM*, 17(1–2):5–32, 2009.

[7] Cataldo Basile, Antonio Lioy, Salvatore Scozzi, and Marco Vallini. Ontology-based policy translation. *CISIS*, 63:117–126, 2009.

[8] Frédéric Cuppens, Nora Cuppens-Boulahia, Thierry Sans, and Alexandre Miège. A formal approach to specify and deploy a network security policy. *FAST*, 173:203–218, 2005.

[9] Andrew Tanenbaum. *Computer Networks (4th Edition)*. Prentice Hall, 2007.

[10] Steven J. Murdoch and Ross Anderson. Tools and technology of Internet filtering. In Ronald Deibert, John Palfrey, Rafal Rohozinski, and Jonathan Zittrain, editors, *Access Denied: The Practice and Policy of Global Internet Filtering*, chapter 3, pages 57–72. MIT Press, 2008.

[11] Jonathan Zittrain and Benjamin Edelman. Empirical analysis of Internet filtering in China. Research Publication 2003-02, The Berkman Center for Internet & Society, 04 2003.

[12] Richard Clayton, Steven J. Murdoch, and Robert N. M. Watson. Ignoring the great firewall of china. In *PET 2006*, pages 20–35. Springer, 2006.

[13] Aldo Gangemi and Valentina Presutti. *Handbook on Ontologies*, chapter Ontology Design Patterns. Springer, 2009.

[14] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley, July 2004.

[15] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*. Wiley, 1996.

[16] Maximillian Dornseif. Government mandated blocking of foreign web content. In *Security, E-Learning, E-Services: Proceedings of the 17. DFN-Arbeitstagung über Kommunikationsnetze*, pages 617–648, 2004.

[17] Bundesrepublik Deutschland. §86 StGB: Verbreiten von Propagandamitteln verfassungswidriger Organisationen, 1975. `http://www.gesetze-im-internet.de/stgb/__86.html` (last accessed: 03/11/14).

[18] OpenNet Initiative. Internet filtering in Saudi Arabia, 2009. `http://opennet.net/research/profiles/saudi-arabia` (last accessed: 03/11/14).

[19] OpenNet Initiative. *Access Controlled: The Shaping of Power, Rights, and Rule in Cyberspace*, pages 369–387. In Deibert et al. [3], 2010.

[20] Bundesrepublik Deutschland. Gesetz zur Bekämpfung der Kinderpornographie in Kommunikationsnetzen. *Bundesgesetzblatt*, (6):78–80, 02 2010. http://www.bgbl.de/banzxaver/bgbl/start.xav?startbk=Bundesanzeiger_BGBl&jumpTo=bgbl110s0078.pdf (last accessed: 03/11/14).

[21] Deutsche Telekom AG. Our code of conduct: The way we work. Booklet, 2012. http://www.telekom.com/code-of-conduct-en (accessed: 29/07/13).

[22] Global Network Initiative. Principles on freedom of expression and privacy, 01 2011. http://www.globalnetworkinitiative.org/principles/index.php (last accessed: 03/11/14).

[23] Ian Sommerville and Gerald Kotonya. *Requirements Engineering: Processes and Techniques*. John Wiley & Sons, Inc, 1998.

[24] Bundesrepublik Deutschland. Die Gesetzgebung des Bundes. In *Grundgesetz für die Bundesrepublik Deutschland*, chapter VII. Bundesrepublik Deutschland, 1949. http://www.gesetze-im-internet.de/gg/ (last accessed: 03/11/14).

[25] Ansgar Scherp, Carsten Saathoff, Thomas Franz, and Steffen Staab. Designing core ontologies. *Applied Ontology*, 6(3):177–221, 2011.

[26] Valentina Presutti and Aldo Gangemi. Content ontology design patterns as practical building blocks for web ontologies. In *IEEE ER 2008*. Springer, 2008.

[27] Aldo Gangemi, Nicola Guarino, Claudio Masolo, Alessandro Oltramari, and Luc Schneider. Sweetening ontologies with DOLCE. In *EKAW 2002*, pages 223–233. Springer, 2002.

[28] W3C OWL Working Group. OWL 2 Web Ontology Language document overview, 10 2009. http://www.w3.org/TR/owl2-overview/ (last accessed: 03/11/14).

[29] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The description logic handbook: theory, implementation, and applications*. Cambridge University Press, 2003.

[30] Aldo Gangemi and Peter Mika. Understanding the semantic web through descriptions and situations. In *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, pages 689–706. Springer, 11 2003.

[31] Rinke Hoekstra, Joost Breuker, Marcello Di Bello, and Alexander Boer. The LKIF core ontology of basic legal concepts. *LOAIT*, 321:43–63, 2007.

[32] Rinke Hoekstra, Joost Breuker, Marcello di Bello, and Alexander Boer. LKIF core: Principled ontology development for the legal domain. *Law, Ontologies and the Semantic Web: Channelling the Legal Information Flood*, 188:21, 2009.

[33] Aldo Gangemi, Maria-Teresa Sagri, and Daniela Tiscornia. A constructive framework for legal ontologies. In V. Richard Benjamins, Pompeu Casanovas, Joost Breuker, and Aldo Gangemi, editors, *Law and the Semantic Web*, volume 3369, pages 97–124. Springer, 2005.

[34] Aldo Gangemi. Design patterns for legal ontology construction. In Pompeu Casanovas, Maria Angela Biasiotti, Enrico Francesconi, and Maria Teresa Sagri, editors, *LOAIT 2007*, pages 65–85, 2007.

[35] Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 07 1948.

[36] Nicodemos Damianou, Naranker Dulay, Emil Lupu, and Morris Sloman. The Ponder policy specification language. In *POLICY 2001*, pages 18–39. IEEE, 01 2001.

[37] Renato Iannella, Susanne Guth, Daniel Pähler, and Andreas Kasten. ODRL version 2.0 core model, 04 2012. http://www.w3.org/community/odrl/two/model/ (last accessed: 03/11/14).

[38] Renato Iannella and Susanne Guth. ODRL version 2.0 common vocabulary, 04 2012. http://www.w3.org/community/odrl/two/vocab/ (last accessed: 03/11/14).

[39] Emil C. Lupu and Morris Sloman. Conflicts in policy-based distributed systems management. *IEEE TSE*, 25(6):852–869, 1999.
[40] Ansgar Scherp, Daniel Eißing, and Carsten Saathoff. A method for integrating multimedia metadata standards and metadata formats with the multimedia metadata ontology. *IJSC*, 6(1):25–49, 2012.
[41] Yilu Zhou, Edna Reid, Jialun Qin, Hsinchun Chen, and Guanpi Lai. US domestic extremist groups on the web: Link and content analysis. *IEEE Intelligent Systems*, 20(5):44–51, 2005.
[42] Declan McCullagh. Google excluding controversial sites. Online news article, 10 2002. `http://news.cnet.com/2100-1023-963132.html` (last accessed: 03/11/14).
[43] Vince Fuller, Tony Li, Jessica Yu, and Kannan Varadhan. Classless inter-domain routing (cidr): an address assignment and aggregation strategy. RFC 1519, Network Working Group, 09 1993. `http://tools.ietf.org/html/rfc1519` (last accessed: 03/11/14).
[44] Jon Postel. Internet control message protocol. RFC 792, Network Working Group, 1981. `http://tools.ietf.org/html/rfc792` (last accessed: 03/11/14).
[45] Tim Bray. An http status code to report legal obstacles. Internet-draft, Network Working Group, 2013. Work in progress. `http://tools.ietf.org/html/draft-tbray-http-legally-restricted-status-03` (last accessed: 03/117/14).
[46] Jaehong Park and Ravi Sandhu. Towards usage control models: Beyond traditional access control. In *SACMAT 2002*, pages 57–64. ACM, 2002.
[47] Ravi Sandhu and Jaehong Park. Usage control: A vision for next generation access control. *Computer Network Security*, 2776:17–31, 2003.
[48] Núria Casellas. Legal ontologies. In *Legal Ontology Engineering*, volume 3 of *Law, Governance and Technology Series*, pages 109–169. Springer, 2011.
[49] Michel Buffa and Catherine Faron-Zucker. Ontology-based access rights management. In *Advances in Knowledge Discovery and Management*, volume 398 of *Studies in Computational Intelligence*, pages 49–61. Springer, 2012.
[50] Brian McBride. RDF schema 1.1, 02 2014. `http://www.w3.org/TR/rdf-schema/` (last accessed: 03/11/14).
[51] Henning Schulzrinne, Hannes Tschofenig, Jr. John B. Morris, Jorge R. Cuellar, James Polk, and Jonathan Rosenberg. Common policy: A document format for expressing privacy preferences. RFC 4745, Network Working Group, 2007. `http://tools.ietf.org/html/rfc4745` (accessed: 29/07/13).
[52] Tim Moses. Privacy policy profile of XACML v2.0. Oasis standard, OASIS, 02 2005. `http://docs.oasis-open.org/xacml/2.0/PRIVACY-PROFILE/access_control-xacml-2.0-privacy_profile-spec-os.pdf` (last accessed: 03/11/14).
[53] Anne Anderson. A Comparison of Two Privacy Policy Languages: EPAL and XACML. Technical report, Sun Microsystems Laboratories, 09 2005.
[54] Hal Abelson, Ben Adida, Mike Linksvayer, and Nathan Yergler. ccrel: The creative commons rights expression language. W3C member submission, Creative Commons, 2008.
[55] Víctor Rodríguez-Doncel, Mari Carmen Suárez-Figueroa, Asunción Gómez-Perez, and María Poveda-Villalón. License linked data resources pattern. In *Proc. of the 4th Int. Workshop on Ontology Patterns*, 2013.
[56] Víctor Rodríguez-Doncel, Mari Carmen Suárez-Figueroa, Asunción Gómez-Perez, and María Poveda-Villalón. Licensing patterns for linked data. In *Proc. of the 4th Int. Workshop on Ontology Patterns*, 2013.
[57] Digital Library Federation. Metadata Encoding and Transmission Standard: Primer and reference manual. Technical report, Digital Library Federation, 2010. `http://www.loc.gov/standards/mets/METSPrimerRevised.pdf` (last accessed: 03/11/14).

[58] Xin Wang. MPEG-21 rights expression language: Enabling interoperable digital rights management. *IEEE Multimedia*, 11(4):84–87, 2004.

[59] International Press Telecommunications Council. RightsML: Specification of a profile and vocabulary for the communication of usage rights in ODRL 2.0. IPTC Standards, 2013. `http://www.iptc.org/std/RightsML/1.1/RightsML_1.1EP2-spec_1.pdf` (last accessed: 03/11/14).

[60] A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni, and J. Lott. KAoS policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement. In *POLICY 2003*, pages 93–96, 2003.

[61] Lalana Kagal, Tim Finin, and Anupam Joshi. A policy language for a pervasive computing environment. In *POLICY 2003*, pages 63–74. IEEE, 2003.

[62] Gianluca Tonti, Jeffrey M. Bradshaw, Renia Jeffers, Rebecca Montanari, Niranjan Suri, and Andrzej Uszok. Semantic web languages for policy representation and reasoning: A comparison of KAoS, Rei, and Ponder. *The Semantic Web*, 2870:419–437, 2003.

[63] Mark R. Brown and Andrew C. Greenberg. On formally undecidable propositions of law: Legal indeterminacy and the implications of metamathematics. *Hastings Law Journal*, 43:1439, 1992.

[64] Bundesrepublik Deutschland. § 202c Vorbereiten des Ausspähens und Abfangens von Daten, 2007. `http://www.gesetze-im-internet.de/stgb/__202c.html` (last accessed: 03/11/14).

[65] Mark M. Seeger. Three years hacker paragraph. *Datenschutz und Datensicherheit*, 34(7):476–478, 2010.

[66] Open Networking Foundation. Sdn architecture. Technical Report SDN ARCH 1.0 06062014, Open Networking Foundation, 06 2014. `https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN_ARCH_1.0_06062014.pdf` (last accessed: 03/11/14).