User Guide of NorJADE Framework

NorJADE is a framework to develop normative multi-agent systems using the very popular **JADE** platform and well known technologies (like **Aspect-Oriented Programming** and **Ontologies**). Hence, we will present the ontology used to create **NorJADE** framework, then we will present the aspects used to enhance **JADE** platform with the normative aspects.

1. NorJADE ontology

Obviously, using norms requires representing them. Hence, we choose ontology to represent the norms. For this reason, we developed an ontology that includes all the important concepts to represent different types of norms. Figure 1 represents the concepts used to specify the norms in NorJADE framework (like: agent, behaviours, deontic operators, regulation mechanism…etc.). Naturally, these concepts are interconnected using several relations (Figure 2). For example, a law is **specified by** a deontic operator (***Obligation***, ***Recommendation*** or ***Prohibition***), **controls** behaviour, **has validity**, **applied by** an enforcer, **implemented by** a regulation mechanism (Regimentation or Enforcement) and **has a consequence** (Reward or Punishment).

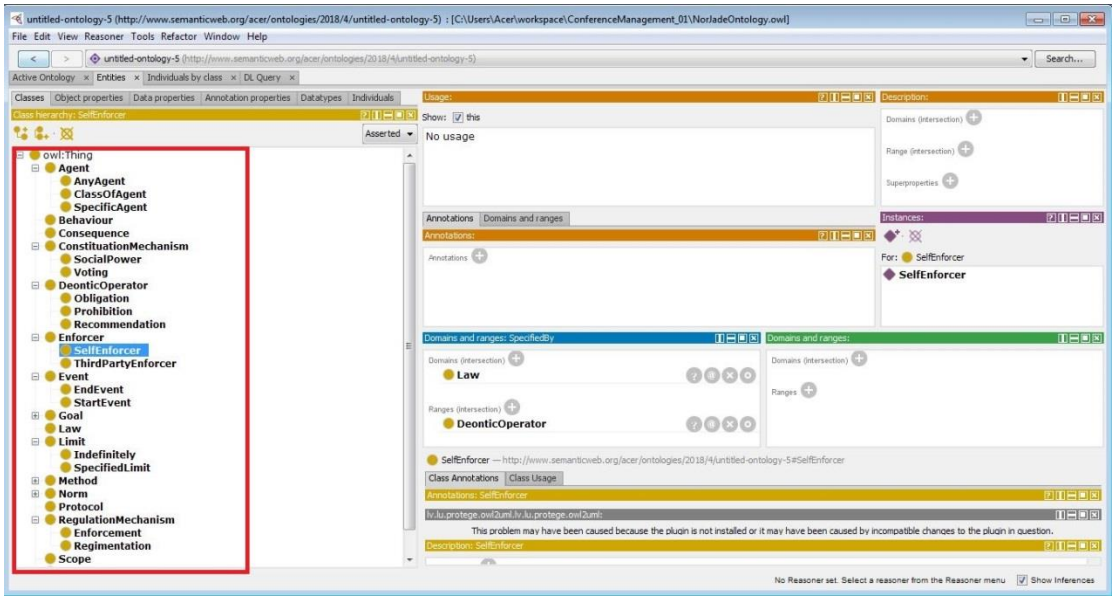

Figure 01: Concepts used to specify norms in NorJADE.

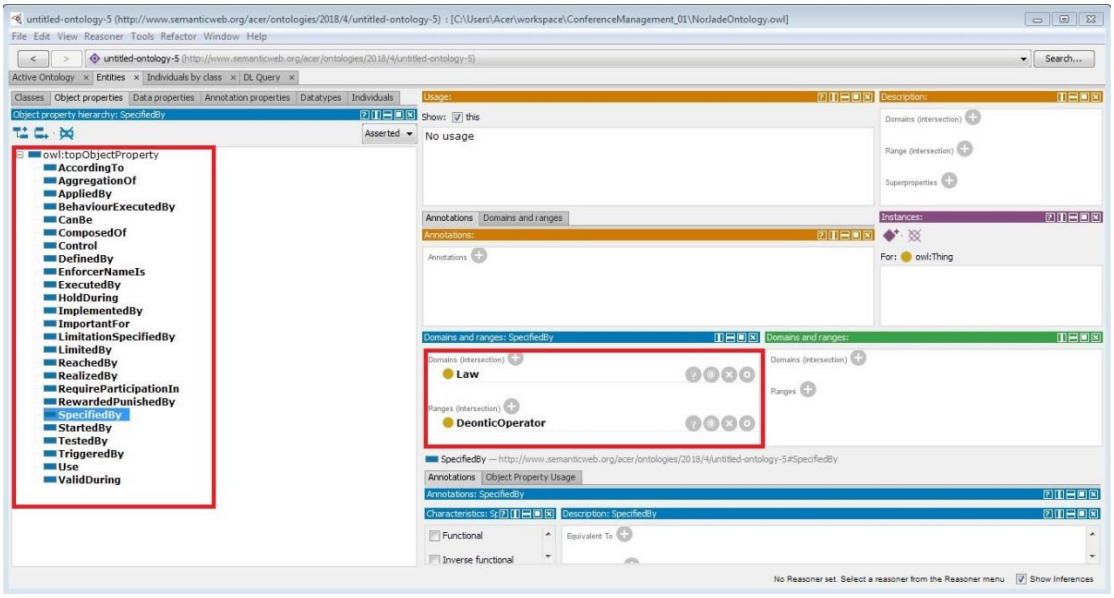

Figure 02: Relations between concepts in NorJADE ontology.

In order to use our framework, you have to create individuals from the existence classes to represent the norms that control your system. For example, to develop a conference management system (code available online), you must create laws that control agents in such system like ***Dispatching Papers' law***, ***Notification Law***, ***Redaction Paper Law*** and ***Submitting Paper Law***.

Figure 03 presents `DispatchingPapersLaw.` This law used the ***obligation deontic*** operator to control the `DispatchingPaper` behaviour (That means it is mandatory to the conference chairman to execute the `DispatchPaper` behaviour during the validity time of this law). The ***validity*** of this law is specified by the `DispatchingValidity` which ***started by*** the deadline and limited by a ***scope*** (10 time unit). If the agent did not execute this behaviour during the validity period of the law, a ***punishment*** will be executed (it is consisted in this case of decreasing the credibility of the conference). This punishment is executed by the ***enforcement mechanism***. This law is applied by ***self-enforcement*** because the conference chairman will decrease its credibility by himself.

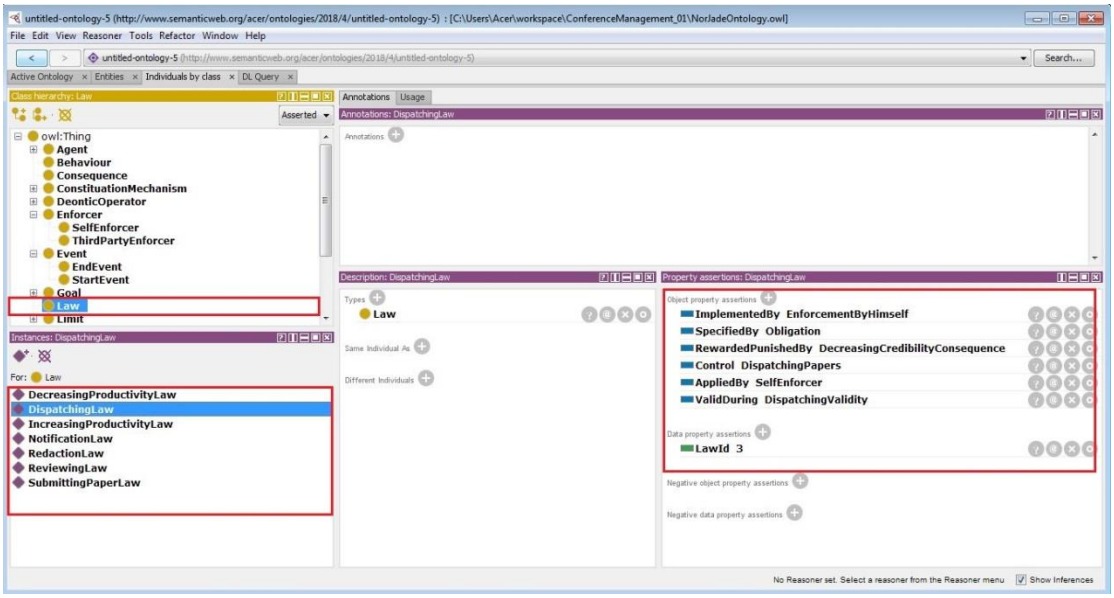

Figure 03: The dispatching paper law represented in NorJADE ontology.

Figure 04 represents another law (the `SubmittingPaperLaw`). According to this law, `SubmittingPaper` behaviour () is ***prohibited*** (**specified by Prohibition**) during the validity time (`ValidDuring` `SubmittingValidity`) which ***started by*** dispatching paper and it is ***indefinitely*** (Naturally, it is not allowed sending papers after deadline and dispatching papers). However, in this case the regulation mechanism is ***regimentation***. Consequently, our framework does not allow the execution of submitting paper behaviour even if the agent attempted this action.

2. NorJADE Aspects

In order to use our framework, the users should use some aspects developed to enhance the **JADE** platform with the normative manipulation capabilities. These aspects are regrouped mainly in two packages: `NorJADEAspects` and `NorJADEontology`. Because **NorJADE** is developed as a framework, the users can reuse all the source code with just negligible changes. Exactly, users should only update `InterceptEvents` aspect to specify the pertinent events according to their application and enhanced the original application with punitions and rewards.

`NorJADEOntology` package contains only one class (called: `NorJADEOntologyBase`) used to manipulate the ontology in order to extract the different information about norms. So, this class is composed of some **SPARQL** requests that are used to manipulate the **NorJADE Ontology** in order to collect information about the specified norms (for example the methods: `getLaw`**,** `getLawProperties`**,** `getEnforcer`**,** `getEventproperties`…etc.). These methods are used later by the different aspects to check the compliance of the executed behaviours with norms and to execute the consequences of these norms.

`NorJADEAspects` package contains several aspects to manipulate norms. First of all, the aspect `ExtendingAgent` is used to enhance the agent with the different functions and structures required to manipulate norms thanks to introduction mechanism. As example of structures used to enhance the standard JADE agent, we specify tow vectors to save all the executed behaviours and intercepted events. Saving the executed behaviours allows checking if an obligate behaviour is executed or not. The intercepted events allow checking the validity of a law. Moreover, we enhanced the standard **JADE** agent with two methods that will execute the consequence of a given law. The first method is applied when the enforcer is self-enforcer (`SelfEnforcementApplication`) and the second method is applied in the case of the third party enforcer (`ThirdPartyEnforcement`).

The second aspect is `ExtendingBehaviour`**.** It is used to extend **JADE** behaviours with some required extensions to abort behaviours according to some specific norms (case of regimentation mechanism where an agent must be prevented to execute a behaviour).

The third aspect (called `InterceptBehaviour`) is used to intercept the execution of a behaviour (the method `action()`) and process this behaviour according to the specified norms. Hence, when we intercept the execution of a behaviour, we should send a request to the ontology with the name of the behaviour to verify if this behaviour is regulated by a norm or not. If the behaviour is regulated by a norm, then we should execute the adequate process (for example, if the behaviour is prohibited with regimentation mechanism, then we should prevent its execution; but if the behaviour is prohibited with the enforcement mechanism, then we should allow its execution with the execution of a punishment).

The aspect `InterceptEvents` is used to intercept the pertinent events related to the norms. Events are used mainly to specify the validity of norms. Hence, when we intercept an event we will process it according to its type. If the intercepted event is a start one, we should save it. On the other hand, if the intercepted event is an end one, we will check if the associated obligate behaviour is executed or not.

The aspect `ListofAgents` is used to save references to all agents created in the project. These references allow manipulating agents mostly when we must execute self-enforcement. In this case the

agent is processed as a simple object because it will be obligate to execute the punishment (i.e the agent will lose its autonomy).

The aspect `LoadNorJADEontology` is used only to load the `NorJADEOntology` when the main method is executed.

In the aspect `ProceduralNorm`, we extend the standard JADE agent with the method `ReachGoal` that allows specifying the required behaviour to reach this goal.

In the aspect `ConstitutiveNorm`, we extend the standard JADE agent with methods that allow adding, suppressing and updating norms.

3. How can we use the framework?

In order to use our framework, you have to :

1. Create an instance of `NorJADEOntology` that reflects the norms of the developed software. For this reason, you have to create individuals for all the classes defined in `NorJADEOntology`.
2. Put the developed ontology in your project directory.
3. Use all the aspects and classes defined in the packages `NorJADEOntology` and `NorJADEAspects` (except `InterceptEvents` aspect) without changing.
4. Update the aspect `InterceptEvents` by specifying the pertinent events for the developed system.
5. Updating the code of the original application by adding procedural and constitutive norms (optional step)
6. Create in the package `RewardPunishment` the behaviours that represent the punishment or reward.

4. Details about packages, aspects, classes and methods

Package	Class / Aspect	Method / structure	description
NorJADEAspects	Event		Class used to define an event and it includes some methods to manipulate an event
	ExtendingAgent	ExecutedBehaviours	A vector to save the executed behaviours by an agent
		InterceptedEvent	A vector used to save the intercepted events
		SelfEnforcementApplication	This method is used to apply a self punishment/reward
		ThirdPartyEnforcement	This method isused to apply a third party punishment /reward
		IsExecutedBehaviour	This method is used to verify either a behaviour is executed or not
		IsConcernedByLaw	This method verify if the agent is concerned by a law or not

		IsValidLaw	This method verify either the law is valid or not
	ExtendingBehaviour		This aspect is used to abort a behaviour that is regulated by a regimentation mechanism
	InterceptBehaviour		This aspect is used to intercept behaviour execution and process it according to the regulation mechanism and the deontic operator
	InterceptEvents		This aspect is used to to intercept events and process them according to their type (start or end type).
	ListOfAgents	Agents	A vector that is used to save references about agents in order to manipulate them (for example, when we need the execution of a method as punishment reward)
	LoadNorJadeOntology		An aspect used to load the NorJadeOntology when main function is executed
	ProceduralNorm		This aspect is used to define procedural norms
	ConstitutiveNorm		This aspect is used to enhanced JADE standard agent by functions that allow adding, updating and suppressing laws
NorJADEOntology	NorJadeOntologyBase	getLaw	Return a law associated to a behaviour
		getLawProperties	Return the properties of a law
		getEnforcer	Return the enforcer
		getAgentClass	Return the class of an agent
		getEventProperties	Return the properties of an event
		getBehaviour	Return a behaviour associated to a goal
		getBehaviourName	Return the name of a behaviour
		getEventId	Return the identifier of an event
		getLawValidity	Return the validity of a law

		getEvent	Return an event
RewardPunishment	Behaviours to define by users		Describe reward or punishment to execute in the case of norms violation

NorJADE's Development Team

For any question about NorJADE, please contact us

Marir.toufik@gmail.com