Explanation and instance illustration of NCKG ontology classes

Type	Class name (entity/ fact/ relation)		Explanation	Instance illustration
entity	Contract_actor		The parties in a contract. “Either party” denotes one of the parties is the subject of a behavior. “Both party” denotes both of the parties are the subject of a behavior.	“Client”; “ProjectManager”; “Contractor”, “Subcontractor”; “Supervisor”; “ProjectManagerAndContractor”, “ContractorOrSubcontractor”; …
	Contract_object		The object that a Contract_actor acts upon. It includes classes such as “Document”, “Information”, “Environment”, …	“Programme”; “site information”; “early warning meeting”; “Contractor’s design”, “defect”, “changeOfScope”.
	Contract_property		The descriptions of the current definition, status, or included content of a Contract_object.	“submitted”; “isNotIdentified”; “isChanged”, “isNotCompensationEvent”.
	Contract_constraint	TimeConstraint	The time or time period during which certain event happens.	“withinOneWeekOfStartingDate; “winthinPeriod for reply”; “beforeKeyDate”.
		AmountConstraint	The required amount, currency of certain Contract_actor conducting a certain Event.	“in the amount and currencies stated in the Contract Data”
		ResultConstraint	The expected result of certain Contract_actor conducting a certain Event.	If an event occurs which stops the Contractor from completing the whole of the works. “the failure of completing whole of the works” is a Result_constraint.
		ConditionConstraint	The precondition of certain Contract_actor conducting a certain Event.	The Contractor may publicise the works only with the Client’s agreement. “Client’s

			agreement” is a ConditionConstraint
entity2entity relation	hasActionTo	“hasActionTo” is the relation connecting a Contract_actor and a Contract_object.	“submit”; “issue”; “revise”; “extend”; “design”; “pay”; “propose to”; “agree to”; “obtain from”; “not allow”; “do not start”; “intend to transfer”; “is responsible for”.
	hasProperty	“hasProperty” is the relation connecting a Contract_object and a Contract_property entity.	“hasProperty” is directly used to connect Object and Property, it has no instances.
fact	Contract_event	The fact of a Contract_actor performs an action. Its instance should be in the form of <Contract_actor, hasActionTo, Contract_object>.	<PM, issue certificate>; <Client, notify, Contractor>; <PMandContractor, agreeTo, extension>.
	Contract_statement	The fact of the description of a Contract_object. Its instance should be in the form of <Contract_object, hasProperty, Contract_property>.	<Programme, hasStatus, isSubmitted>; <communication, hasStatus, hasEffect>.
entity2fact relation	hasConstraint	“hasConstraint” is the relation connecting an Event and a Constraint.	“hasTimeConstraint”; “hasAmountConstraint”; “hasResultConstraint”; “hasConditionConstraint”; One of the example triples: <Event, hasTimeConstraint, beforeReplyDueDate>. The Event could be <ProjectManger, agreeTo, extension>.