Walkthrough Supporting C2SIM Ontologies

Magdalena Dechand
Fraunhofer Institute for Communication, Information Processing and Ergonomics FKIE
GERMANY

magdalena.dechand@fkie.fraunhofer.de

ABSTRACT

The Command and Control Systems – Simulation Systems Interoperation (C2SIM) international standard specifies an important Modelling and Simulation (M&S) standard. It defines content for information interchange across Command and Control (C2) systems, simulation systems, and Robotic and Autonomous Systems (RAS) [13]. The standard was developed by the Simulation Interoperability Standards Organization (SISO) and approved in 2020. NATO Modelling and Simulation Group 211 (MSG-211) developed a Research Technical Course titled "Modelling and Simulation Standards in NATO Federated Mission Networking". This Educational Notes paper presents the course content for the topic "Walkthrough supporting C2SIM ontologies" [8]. The paper presents the structure and content of C2SIM Core Logical Data Model, the Standard Military Extension (SMX) and the Land Operation Extension (LOX) ontologies including ontology specific features in Protégé. This overview is to help understand and use the ontologies for own applications. It describes how to model ontology extensions if additional requirements emerge, e.g. for different domains. It also shows the process how to transform the ontologies to an XML schema and produce C2SIM messages to exchange information between systems.

1.0 C2SIM INFORMATION MODEL

C2SIM enables information exchange between a system of systems comprising C2, simulation systems and robotic and autonomous systems. The purpose of this information is to initialize and synchronize the systems and to send, receive and process C2SIM messages such as orders and reports (see Figure 1).

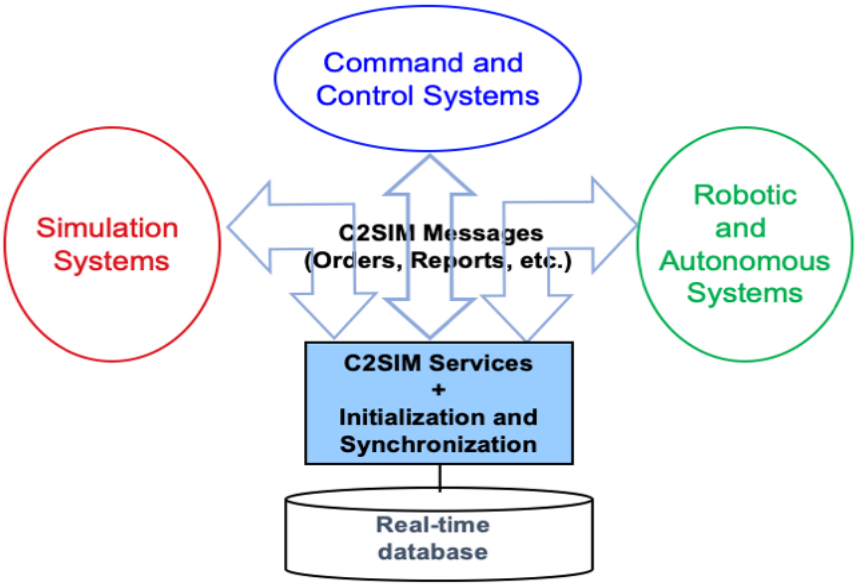

Command and Control Systems
Simulation Systems
Robotic and Autonomous Systems
C2SIM Messages (Orders, Reports, etc.)
C2SIM Services + Initialization and Synchronization
Real-time database

Figure 1: C2SIM System-of-Systems [16].

C2SIM information is exchanged in the form of XML messages derived from a reference knowledge model [6]. This reference model consists of multiple ontology layers built on one another [7]. The Core Logical Data Model as initial layer covers core concepts and represents together with the Standard Military Extension (SMX) basic information that most C2 and simulation systems have in common. The Land Operation Extension (LOX) is the only standardized domain model covering knowledge from the Land Domain (see Figure 2) [13], [15]. It serves as model for the C2SIM extension process.

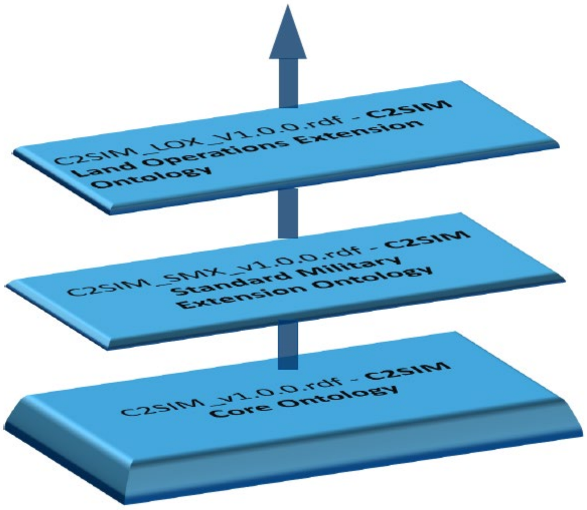

C2SIM_LOX_v1.0.0.rdf - C2SIM
Land Operations Extension
Ontology
C2SIM_SMX_v1.0.0.rdf - C2SIM
Standard Military
Extension Ontology
C2SIM_v1.0.0.rdf - C2SIM
Core Ontology

Figure 2: C2SIM ontology layers.

1.1 C2SIM Core Logical Data Model

The C2SIM Core Logical Data Model (LDM) builds the basis for C2SIM information exchange. The core ontology provides the basic semantic structure and information for C2SIM coalitions. The structure depends on three concepts or ontology classes: *InitializationConcept*, *MessageConcept*, and *C2SIMContent*. *InitializationConcept* provides classes and further ontology features that are necessary to initialize a scenario. *MessageConcept* collects concepts to create messages for Tasking and Reporting as well as *SystemMessages*. Both classes refer to *C2SIMContent* classes that represent general semantic concepts like *AbstractObject*, *Action*, *PhysicalConcept*, *Code*, *Entity*, *EntityType*, *EntityDescriptor*, *EntityState*, *Relationship* and *Resource* (see Figure 3).

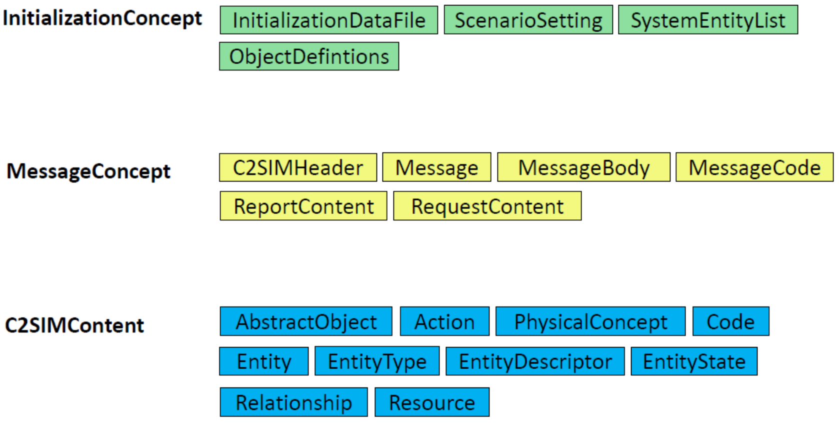
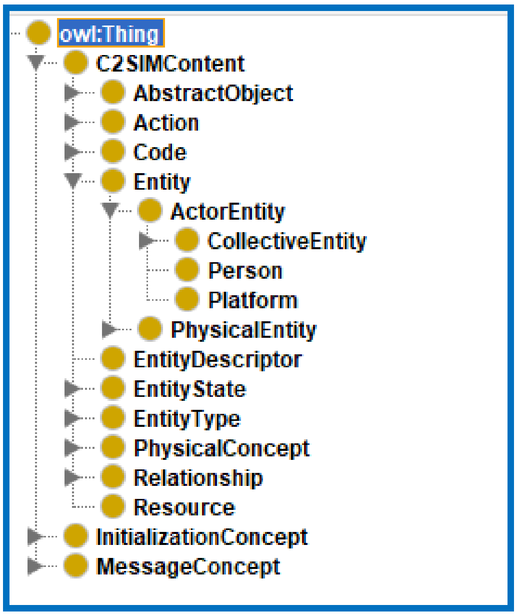


Figure 3: Basic C2SIM concepts/classes.

Classes are organized in a taxonomy meaning they may have hypernymy and hyponymy relations to each other. Hypernymy is the transitive superordination of one class to another, e.g. *ActorEntity* as subclass has a "is a"-relation to its superclass *Entity* but not vice versa. Hyponymy instead refers to the subordination to another class, i.e. *Entity* has the subclasses *PhysicalEntity* and *ActorEntity* that again has the subclass *CollectiveEntity*. Figure 4 reveals the taxonomy view in the ontology editor Protégé [15] which can be expanded at the arrows to show more subclasses in indented lines.



- owl:Thing
  - C2SIMContent
    - AbstractObject
    - Action
    - Code
    - Entity
      - ActorEntity
        - CollectiveEntity
        - Person
        - Platform
      - PhysicalEntity
    - EntityDescriptor
    - EntityState
    - EntityType
    - PhysicalConcept
    - Relationship
    - Resource
  - InitializationConcept
  - MessageConcept

Figure 4: Entity and its subclasses

Apart from the taxonomy relations, ontologies allow to define class attributes. Those are inherited by their respective subclasses. Thus, they correlate with the taxonomy. Subclasses distinguish from its superclasses and sister classes by specifying with additional attributes or property restrictions. Accordingly, *Entity* has subclasses like *ActorEntity* and *CollectiveEntity*. Besides, *Entity* has the property restrictions *hasEntityType min 1 EntityType*, *hasName max 1 xsd:string* and *hasUUIDBase exactly 1 UUIDBase* (see Figure 5). The property restrictions state conditions an instance must fulfil to be regarded as member of that class.

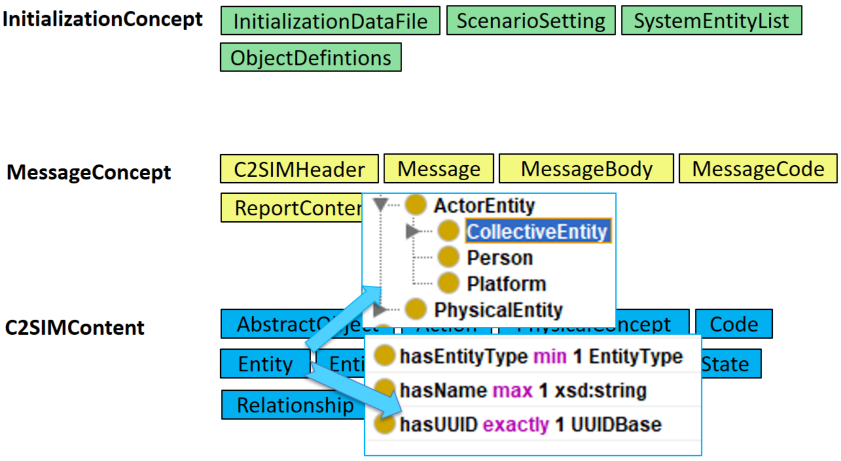
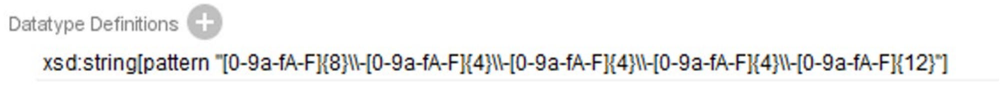


Figure 5: Entity, its subclasses and its attributes.

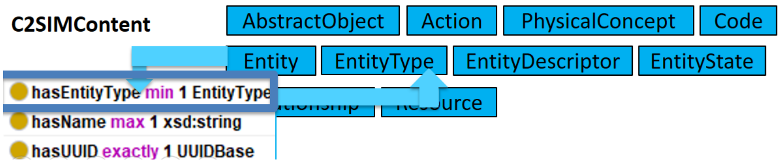
These property restrictions refer either to object properties or datatype properties. Datatype properties assign a value type to a member of a class bearing that property. An *Entity* has an optional datatype property restriction which is *hasName max 1 xsd:string* restricting its name to a string. It also has the datatype property restriction *hasUUID exactly 1 UUIDBase* defining a unique identifier as *UUIDBase. UUIDBase* is a value type or pattern deliberately specified by C2SIM experts using regular expressions (see Figure 6).



```
xsd:string[pattern "[0-9a-fA-F]{8}\\-[0-9a-fA-F]{4}\\-[0-9a-fA-F]{4}\\-[0-9a-fA-F]{4}\\-[0-9a-fA-F]{12}"]
```

Figure 6: Defining Datatypes for datatype properties.

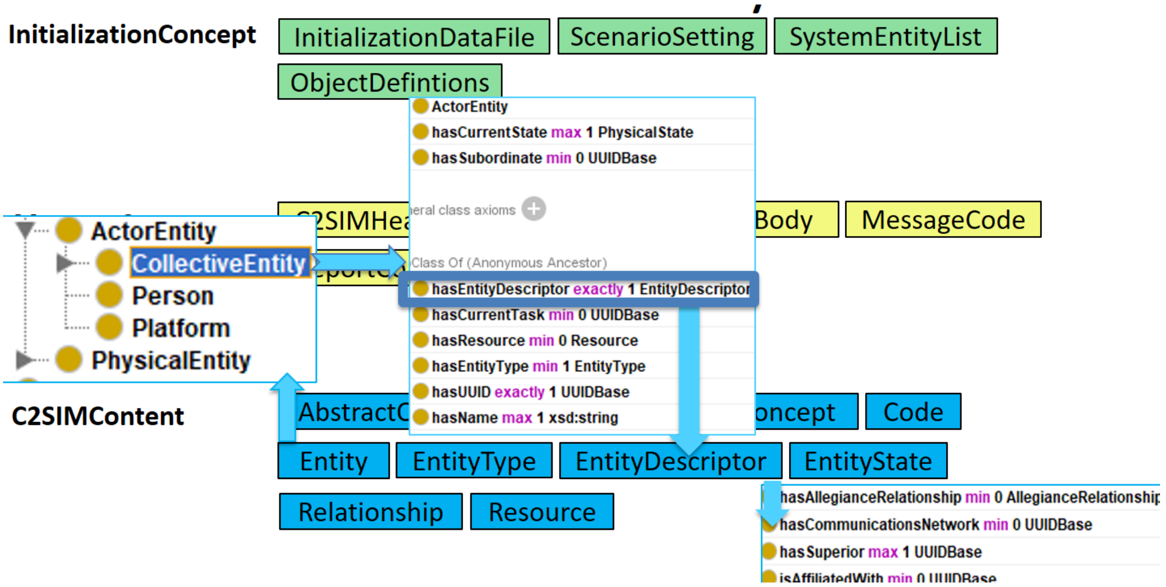
Object property restrictions on the other hand represent attributes linking one class (or its instances) to another while not being super- or subordinated to each other according to the taxonomy. An *Entity* is specified with the object property restriction *hasEntityType min 1 EntityType* relating *Entity* to at least one *EntityType* (Figure 7). As *EntityType* comes with more attributes, those affect *Entity*, too.



hasEntityType min 1 EntityType
hasName max 1 xsd:string
hasUUID exactly 1 UUIDBase

Figure 7: Entity and EntityType linked by an object property restriction.

CollectiveEntity as subsubclass of *Entity* and subclass of *ActorEntity* inherits their property restrictions *(see Class Of (Anonymous Ancestor) in Figure 8)* and specifies by further property restrictions. Accordingly, the property restrictions *EntityDescriptor exactly 1 EntityDescriptor*, *hasCurrentTask min 0 UUIDBase* and *hasRecource min 0 Resource* are inherited from *ActorEntity*. The first property restriction refers to the class *EntityDescriptor* that again refers to an *AllegianceRelationship*, an *Affiliation*, a *Superior* and a *CommunicationNetwork*. The other three property restrictions are inherited from *Entity*.



ActorEntity
hasCurrentState max 1 PhysicalState
hasSubordinate min 0 UUIDBase

hasEntityDescriptor exactly 1 EntityDescriptor
hasCurrentTask min 0 UUIDBase
hasResource min 0 Resource
hasEntityType min 1 EntityType
hasUUID exactly 1 UUIDBase
hasName max 1 xsd:string

hasAllegianceRelationship min 0 AllegianceRelationship
hasCommunicationsNetwork min 0 UUIDBase
hasSuperior max 1 UUIDBase
isAffiliatedWith min 0 UUIDBase

Figure 8: Entity and EntityDescriptor.

The property restrictions *hasCurrentState max 1 PhysicalState* and *hasSubordinate min 0 UUIDBase* (see Figure 9) are specific to *CollectiveEntity*. The object property *hasCurrentState* associates the class to *PhysicalState,* a subclass of *EntityState* including *Location*, *Orientation*, *Health*, *Speed* of an *Entity*.

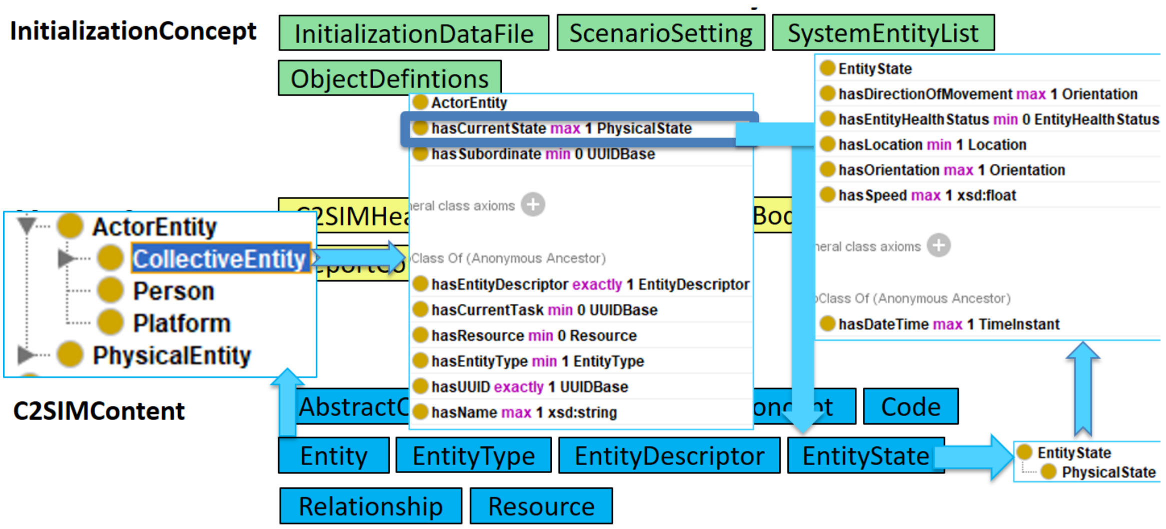

InitializationConcept
InitializationDataFile
ScenarioSetting
SystemEntityList
ObjectDefintions
ActorEntity
hasCurrentState max 1 PhysicalState
hasSubordinate min 0 UUIDBase
eral class axioms
Class Of (Anonymous Ancestor)
hasEntityDescriptor exactly 1 EntityDescriptor
hasCurrentTask min 0 UUIDBase
hasResource min 0 Resource
hasEntityType min 1 EntityType
hasUUID exactly 1 UUIDBase
hasName max 1 xsd:string
Entity State
hasDirectionOfMovement max 1 Orientation
hasEntityHealthStatus min 0 EntityHealthStatus
hasLocation min 1 Location
hasOrientation max 1 Orientation
hasSpeed max 1 xsd:float
eral class axioms
Class Of (Anonymous Ancestor)
hasDateTime max 1 TimeInstant
ActorEntity
CollectiveEntity
Person
Platform
PhysicalEntity
C2SIMHea
Boc
C2SIMContent
AbstractC
oncept
Code
Entity
EntityType
EntityDescriptor
EntityState
Relationship
Resource
Entity State
PhysicalState

Figure 9: Entity and hasCurrentState referring to PhysicalState.

All those *Entity*-related concepts or other *C2SIMContent* concepts are allowed to be used by *InitializationConcept* and *MessageConcept*. For the initialization of systems, C2SIM provides four superclasses of *InitializationConcept* (see green boxes in Figure 10). *InitializationDataFile* allows to describe a terrain, *ScenarioSetting* to specify the setting of a scenario or a *SystemEntityList* to list entities that reside within the systems. *ObjectDefinitions* define objects with object property restrictions referring to *AbstractObject*, *Action* or *Entity* which are subclasses of *C2SIMContent* (see Figure 10).

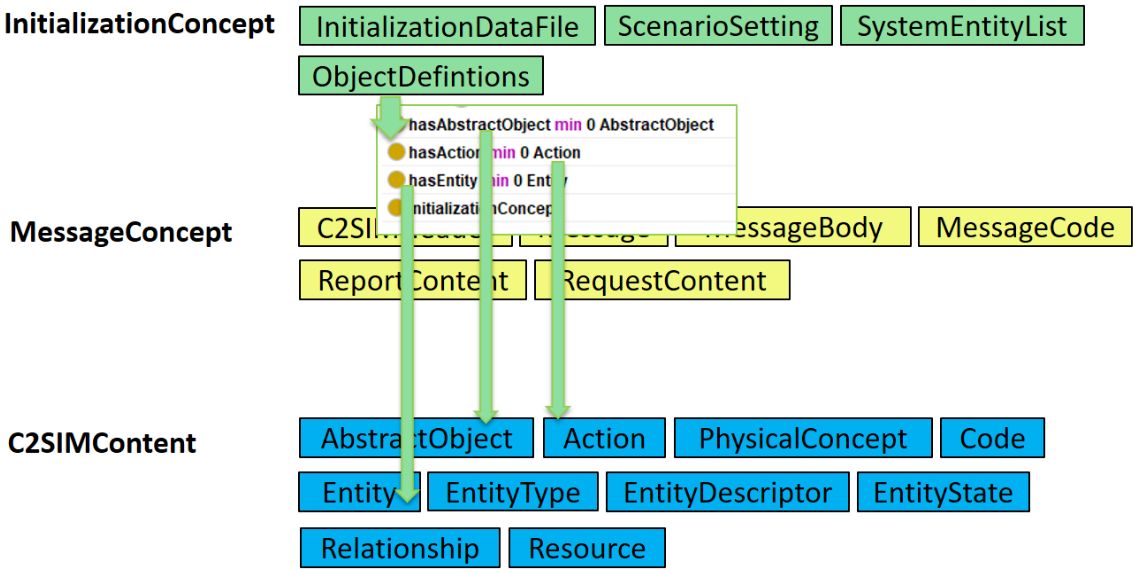


Figure 10: ObjectDefinitions referring to AbstractObject, Action and Entity.

MessageConcept allows the systems to send, receive and process a *SystemMessage* and tasking and reporting messages (see yellow boxes in Figure 11). It covers structural information that messages consist of, i.e. a *Message hasC2SIMHeader exactly 1 C2SIMHeader* and *hasMessageBody exactly 1 MessageBody*. *MessageBody* has the subclass *DomainMessageBody*, which unfolds into the subclasses *AcknowledgementBody*, *OrderBody*, *ReportBody or RequestBody.* They all have a *Sender* and a *Receiver* reference.

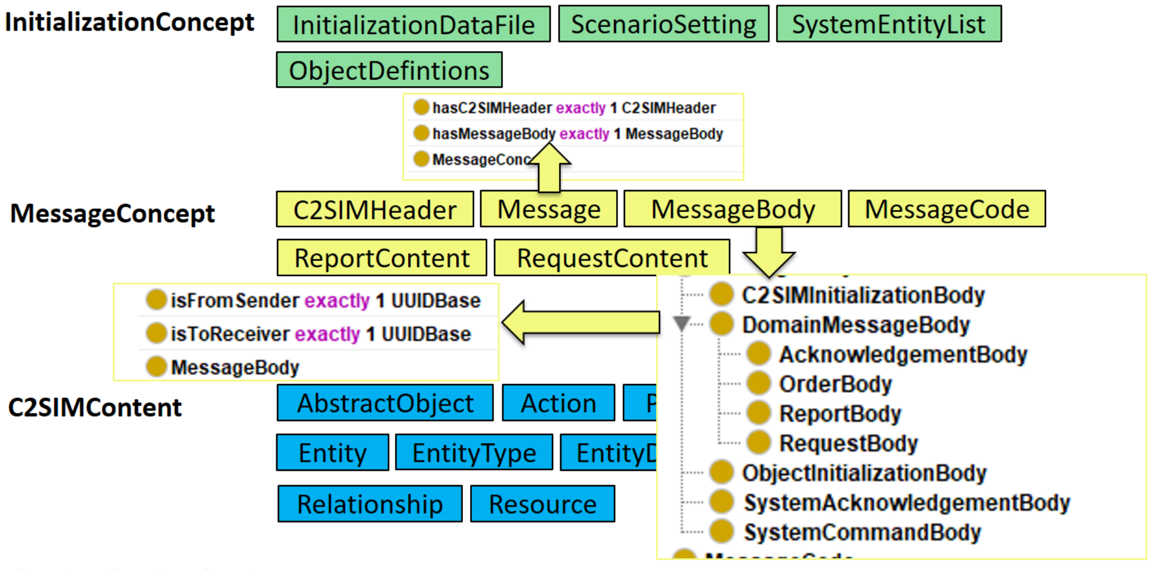


Figure 11: DomainMessages for Tasking and Reporting.

ReportBody inherits the property restrictions *isFromSender exactly 1 UUIDBase* and *isToReceiver exactly 1 UUIDBase* from its superclass *DomainMessageBody.* It further specifies by *hasReportContent min 1 ReportContent*, *hasReportID exactly 1 UUIDBase* and *hasReportingEntity exactly 1 UUIDBase*. *PositionReportContent* and *TaskStatus* are subclasses of *ReportContent* (see Figure 12).

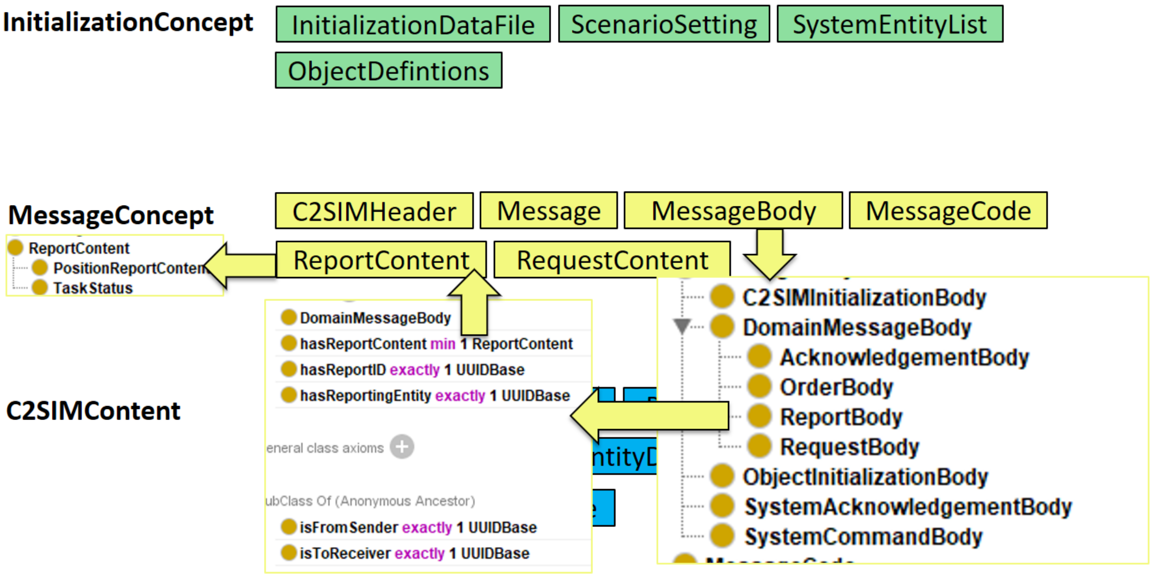


Figure 12: ReportContent as part of ReportBody.

OrderBody is crucial for Orders referring, among others, to *Task* with the object property restriction *hasTask min 0 Task*. *Task* is a subclass of *Action* and thus of *C2SIMContent* (see Figure 13).

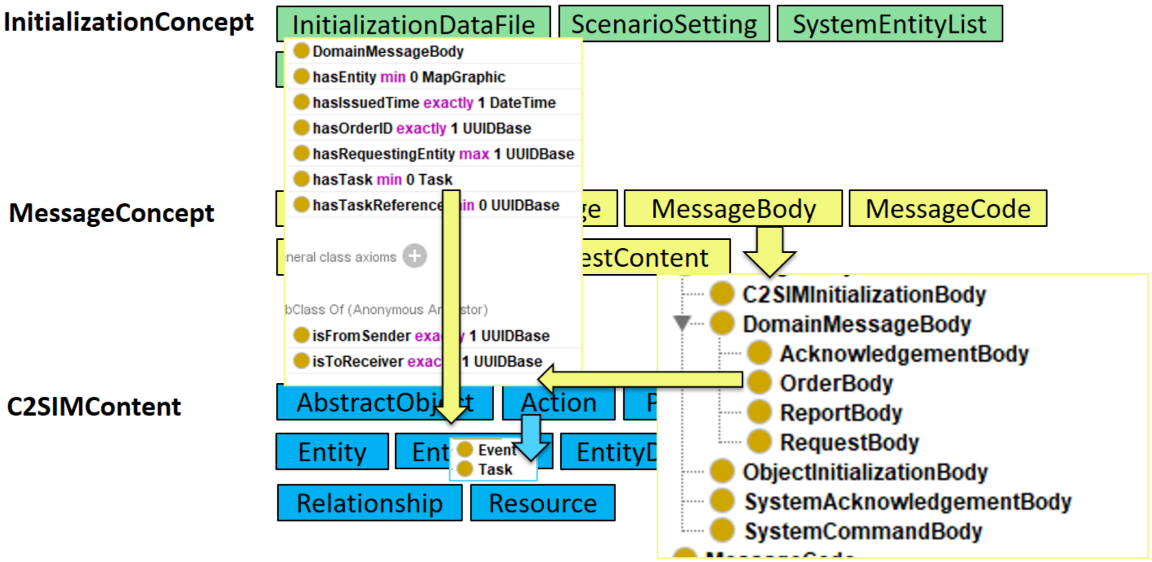


Figure 13: OrderBody with reference to Task.

RequestBody is associated, among others, to *RequestContent* with the object property restriction *hasRequestContent exactly 1 RequestContent.* Its only subclass *TaskRequestContent,* is linked to *Task*. This is similar to *ReportContent* but differs having a distinct command structure. *AcknowledgmentBody* is only specified by the property restriction *hasAcknowledgementTypeCode* referring to *AcknowledgmentCode* which contains the instances *ACKSUCC* or *ACKFAIL*.

1.2 Standard Military Extension

The Standard Military Extension (SMX) is built using the core's structure, classes and property restrictions and adding more military concepts to it. It is extended by one additional class to the main taxonomy structure and comes with further property restrictions. *C2SIMContent* has the subclass *Observation* with more subclasses such as *ActivityObservation, HealthObservation*, *LocationObservation, NameObservation, RecourceObservation*, *SubjectTypeObservation*. They refer to a specific *Entity* or rather *ActorReference* and involve information about its *Activity*, *Health*, *Location* etc. Moreover, *ObservationReportContent* is enclosed as an additional subclass of *ReportContent.* It refers to *Observation* with the help of the new property restriction *hasObservation min 1 Observation* (see Figure 14).

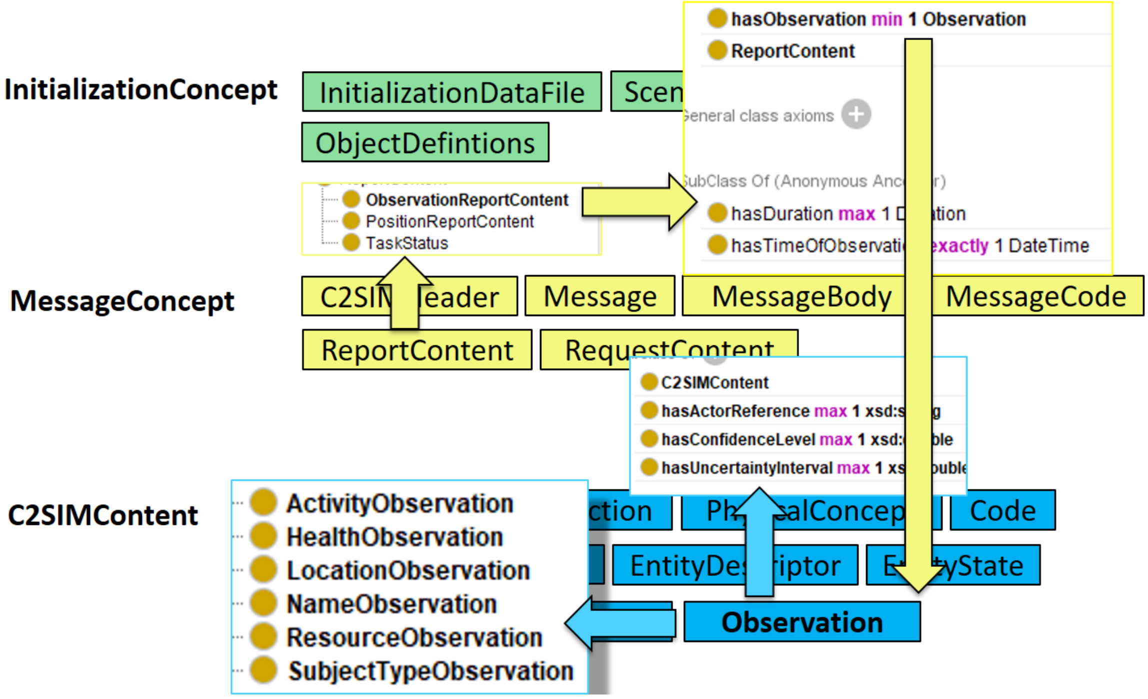


Figure 14: ObservationReportContent with reference to Observation.

RequestContent has the subclass *MIPRequestContent* in the SMX and utilizes the property restriction *hasMIPRequestCategoryCode exactly 1 MIPCategoryCod*e. The class *MIPCategoryCode* as subclass of *MessageCode,* uses codes derived from MIP (Multilateral Interoperability Program) (see Figure 15), a standard for distributed C2 systems.

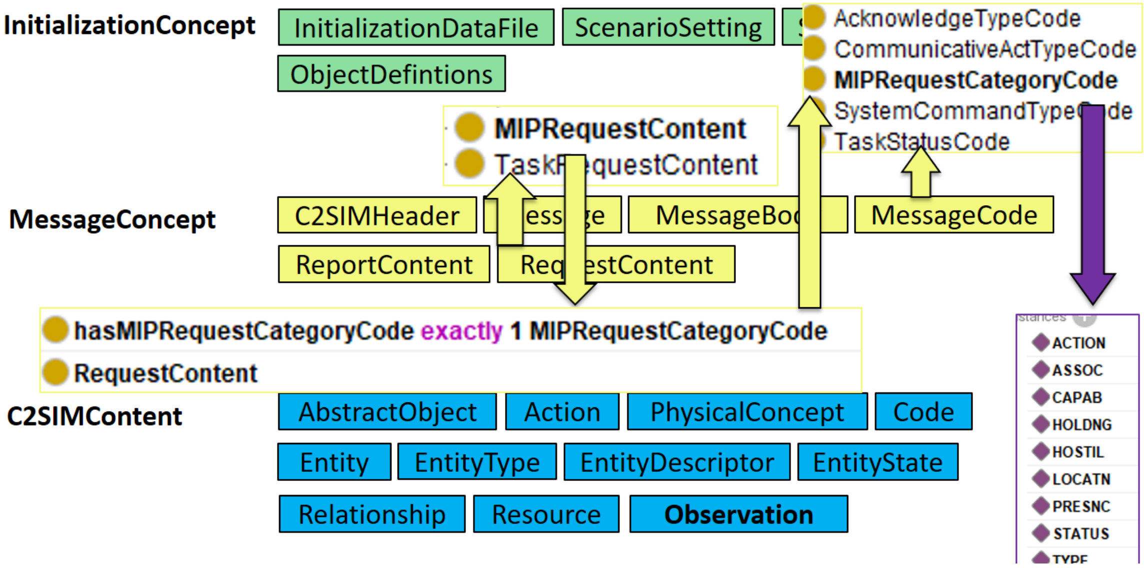


Figure 15: MIPRequestContent using MIPRequestCategoryCode.

Furthermore, SMX offers new subclasses of *CollectiveEntity,* such as *MilitaryOrganiation* and *Unit*. Apart from inheriting restrictions from its superclasses, they are defined by the property restrictions *hasCommandRelation min 0 CommandRelation* and *hasEchelonCode exactly 1 EchelonCode* referring to military specific attributes. *EchelonCode* as subclass of *Code* includes a list of instances containing ARMY, BDE or COY (see Figure 16).

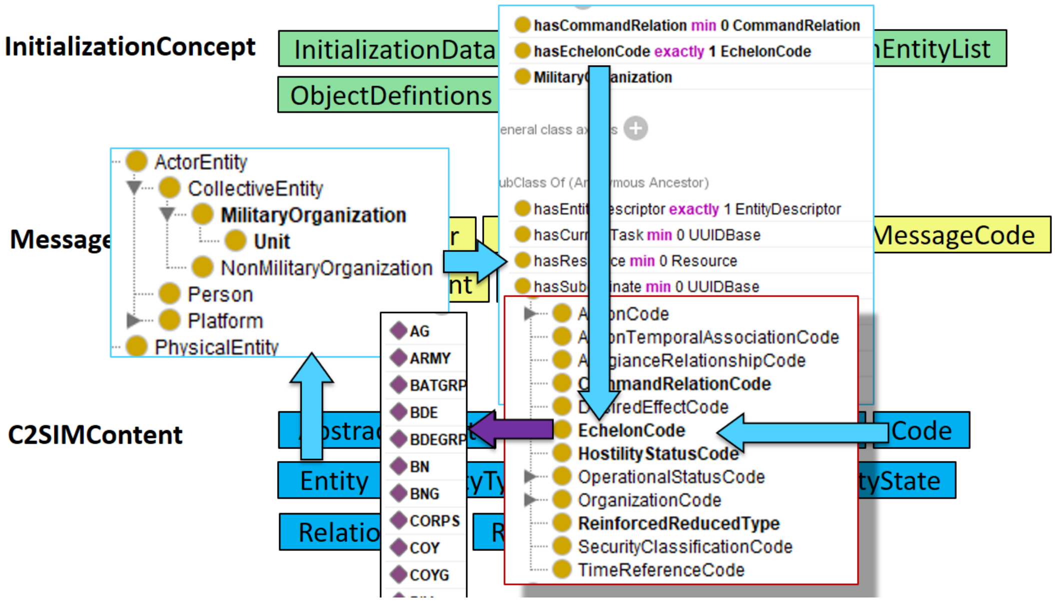


Figure 16: Unit defined by EchelonCode.

A more hidden property restriction is added to further specify *Entity* with *hasEntityDesciptor exactly 1 EntityDescriptor* that again has the new property restriction *hasSide max 1 UUIDBase* (see Figure 17).

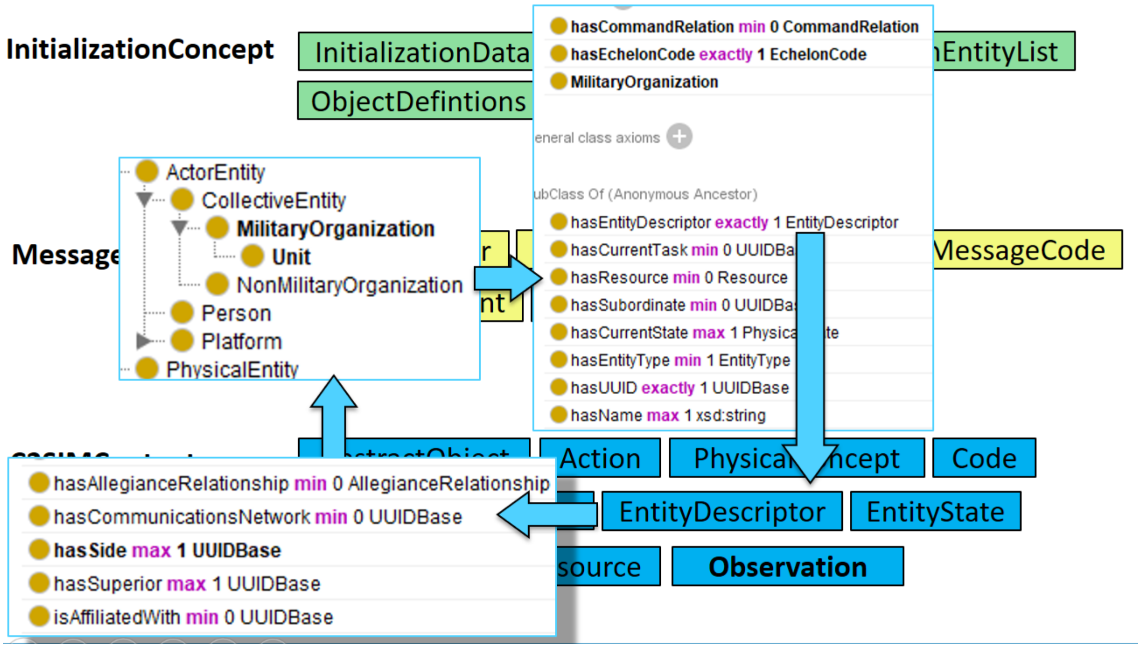

InitializationConcept
InitializationData
ObjectDefintions
nEntityList
Message
MessageCode
hasCommandRelation min 0 CommandRelation
hasEchelonCode exactly 1 EchelonCode
MilitaryOrganization
eneral class axioms
ubClass Of (Anonymous Ancestor)
hasEntityDescriptor exactly 1 EntityDescriptor
hasCurrentTask min 0 UUIDBase
hasResource min 0 Resource
hasSubordinate min 0 UUIDBase
hasCurrentState max 1 PhysicalState
hasEntityType min 1 EntityType
hasUUID exactly 1 UUIDBase
hasName max 1 xsd:string
ActorEntity
CollectiveEntity
MilitaryOrganization
Unit
NonMilitaryOrganization
Person
Platform
PhysicalEntity
AbstractObject
Action
PhysicalConcept
Code
EntityDescriptor
EntityState
source
Observation
hasAllegianceRelationship min 0 AllegianceRelationship
hasCommunicationsNetwork min 0 UUIDBase
hasSide max 1 UUIDBase
hasSuperior max 1 UUIDBase
isAffiliatedWith min 0 UUIDBase

Figure 17: Unit with reference to EntityDescriptor and added hasSide-restriction.

ForceSide as a new subclass of *AbstractObject* is linked to *ForceSideRelation* as subclass of *Relation* (see Figure 18).

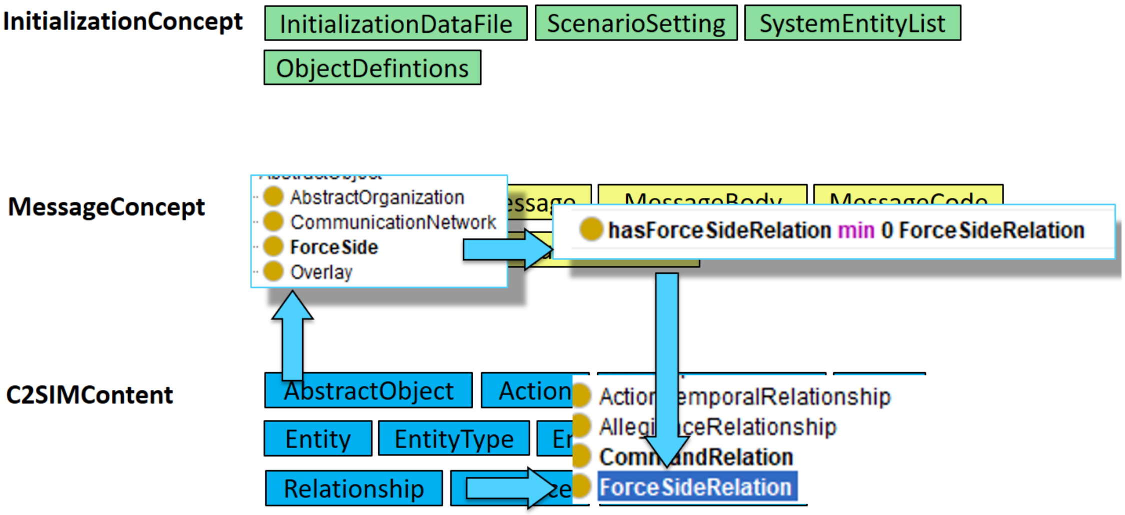


Figure 18: ForceSide as subclass of AbstractObject.

1.3 Land Operation Extension

The land operation extension (LOX) is a third and optional ontology layer. It partly covers the military land operations aspects of Military Scenario Description Language (MSDL) and Coalition – Battle Management Language (BML) that are used by the international land military simulation community. LOX adds four subclasses to *C2SIMContent: PlanPhase, PlanPhaseTrigger, RuleOfEngagement. PlanPhase* is referred by *PlanBody* as subclass of *DomainMessageBody* (Figure 19).

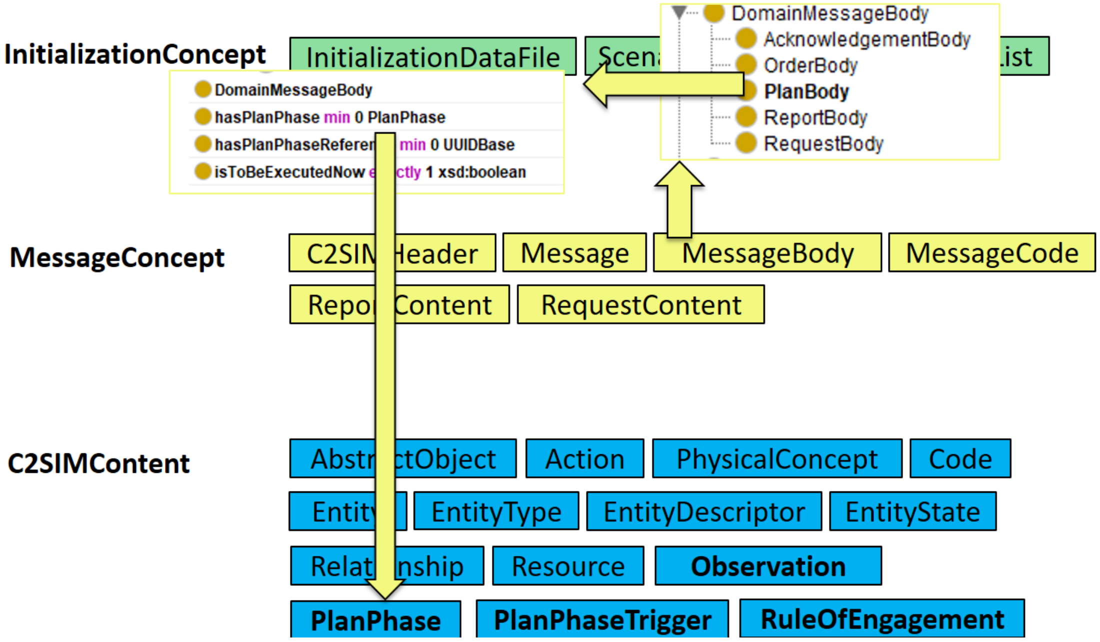


Figure 19: PlanBody, PlanPhase, PlanPhaseTrigger and RuleOfEngagement.

LOX also specifies *Task* with a new subclass: *ManeuverWarefareTask* which can be used for *Orders* and *Requests*. It uses a property restriction *hasRuleofEngagement min 0 RuleOfEngagement* defining the conditions under which weapons are allowed to be used to engage other forces (see Figure 20). The object property restriction *hasTaskFunctionalRelation min 0 TaskFunctionalRelation* refers to a functional relationship between tasks.

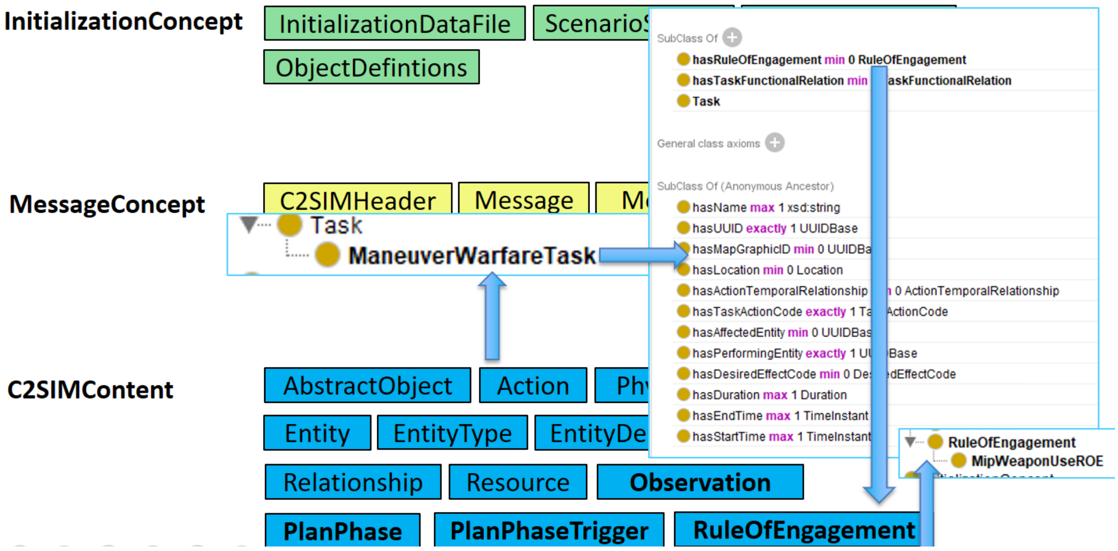
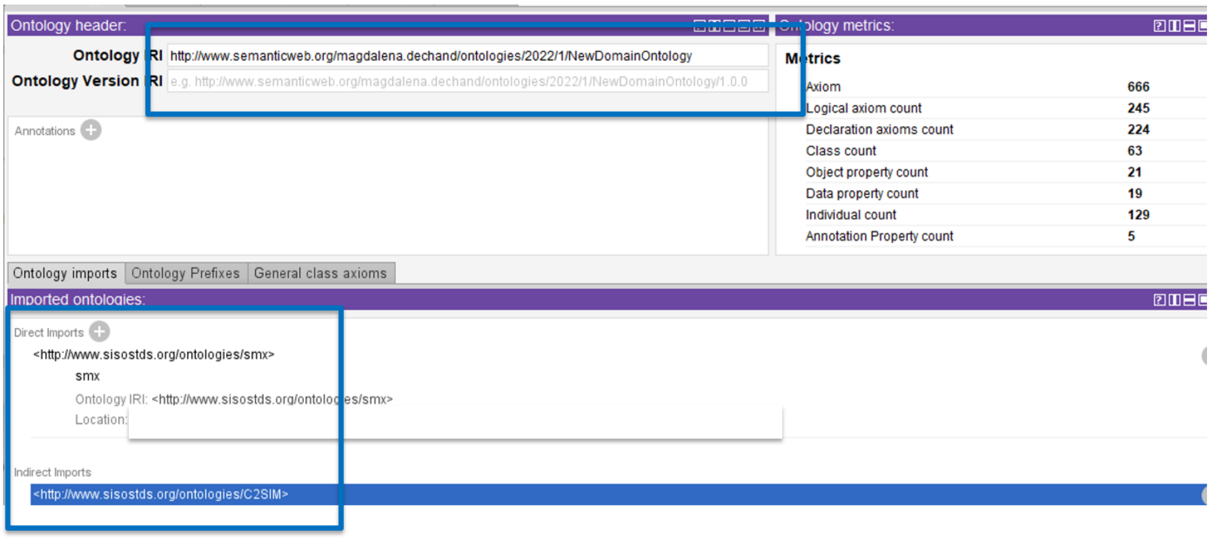

InitializationConcept
InitializationDataFile
Scenario
ObjectDefintions
MessageConcept
C2SIMHeader
Message
M
Task
ManeuverWarfareTask
C2SIMContent
AbstractObject
Action
Ph
Entity
EntityType
EntityDe
Relationship
Resource
Observation
PlanPhase
PlanPhaseTrigger
RuleOfEngagement
SubClass Of
hasRuleOfEngagement min 0 RuleOfEngagement
hasTaskFunctionalRelation min askFunctionalRelation
Task
General class axioms
SubClass Of (Anonymous Ancestor)
hasName max 1 xsd:string
hasUUID exactly 1 UUIDBase
hasMapGraphicID min 0 UUIDBa
hasLocation min 0 Location
hasActionTemporalRelationship 0 ActionTemporalRelationship
hasTaskActionCode exactly 1 Ta ActionCode
hasAffectedEntity min 0 UUIDBas
hasPerformingEntity exactly 1 U Base
hasDesiredEffectCode min 0 De dEffectCode
hasDuration max 1 Duration
hasEndTime max 1 TimeInstant
hasStartTime max 1 TimeInstant
RuleOfEngagement
MipWeaponUseROE

Figure 20: ManeuverWarfareTask and RuleOfEngagement.

2.0 C2SIM – EXTENSION PROCESS

C2SIM not only provides the standard ontology layers Core, Standard Military Extension (SMX) and Land Operation Extension (LOX). It also provides a process to extend the ontologies due to own requirements which may result from the employment of different systems (e.g. uncrewed systems) or different domains (e.g. maritime, cyber etc.) [3], [4], [9]. An extension should be modelled similarly to LOX. It should be built as additional layer on top of at least the Core ontology and the SMX layer using Protégé. Moreover, the C2SIM Ontology Subgroup (COS) manages and maintains the standard. Before or after creating an extension, the COS can help and contact another subgroup dealing with the same domain extension. On the other hand, change requests and problem reports can be submitted to the COS. These are gathered and might lead to adjustments to the standard.

After gathering requirements for a new C2SIM ontology extension, the modelling begins after defining a new namespace (Ontology IRI) and ontology imports (see Figure 21). The imports are crucial to be able to use the information already modelled in the Core and SMX. In case SMX is used, it needs to be set and automatically imports the core ontology.



Ontology header:

Ontology IRI http://www.semanticweb.org/magdalena.dechand/ontologies/2022/1/NewDomainOntology

Ontology Version IRI e.g. http://www.semanticweb.org/magdalena.dechand/ontologies/2022/1/NewDomainOntology/1.0.0

Annotations

Ontology metrics:

Metrics

Axiom	666
Logical axiom count	245
Declaration axioms count	224
Class count	63
Object property count	21
Data property count	19
Individual count	129
Annotation Property count	5

Ontology imports | Ontology Prefixes | General class axioms

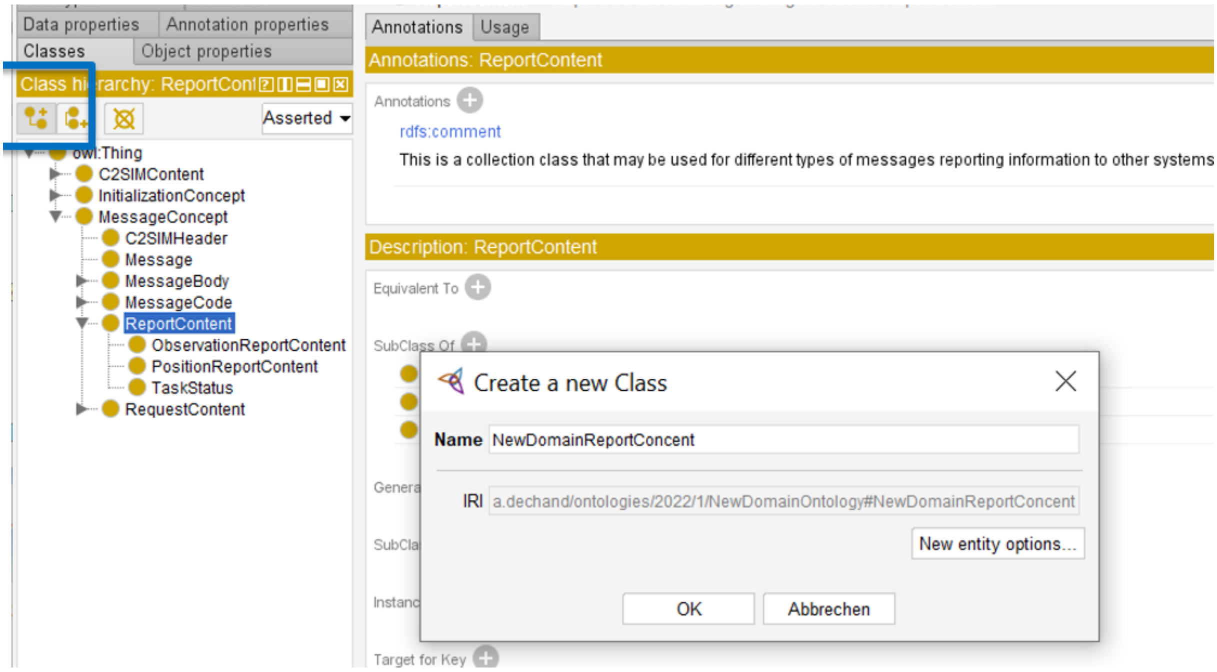
Imported ontologies:

Direct Imports
<http://www.sisostds.org/ontologies/smx>
smx
Ontology IRI: <http://www.sisostds.org/ontologies/smx>
Location:

Indirect Imports
<http://www.sisostds.org/ontologies/C2SIM>

Figure 21: Defining a namespace for ontology extension in Protégé.

Following the structure of the imported ontology layers, features can be added and automatically belong to the extension now. Those features include new subclasses, datatypes, data properties, object properties, property restrictions and instances. Subclasses can extend the taxonomy by expanding a class and creating another (see Figure 22). It automatically will inherit its superclasses' properties.



Data properties | Annotation properties
Classes | Object properties

Class hierarchy: ReportContent

Asserted

- owl:Thing
  - C2SIMContent
  - InitializationConcept
  - MessageConcept
    - C2SIMHeader
    - Message
    - MessageBody
    - MessageCode
    - ReportContent
      - ObservationReportContent
      - PositionReportContent
      - TaskStatus
    - RequestContent

Annotations | Usage

Annotations: ReportContent

Annotations
rdfs:comment
This is a collection class that may be used for different types of messages reporting information to other systems

Description: ReportContent

Equivalent To

SubClass Of

Genera

SubCla

Instanc

Target for Key

Create a new Class

Name NewDomainReportConcent

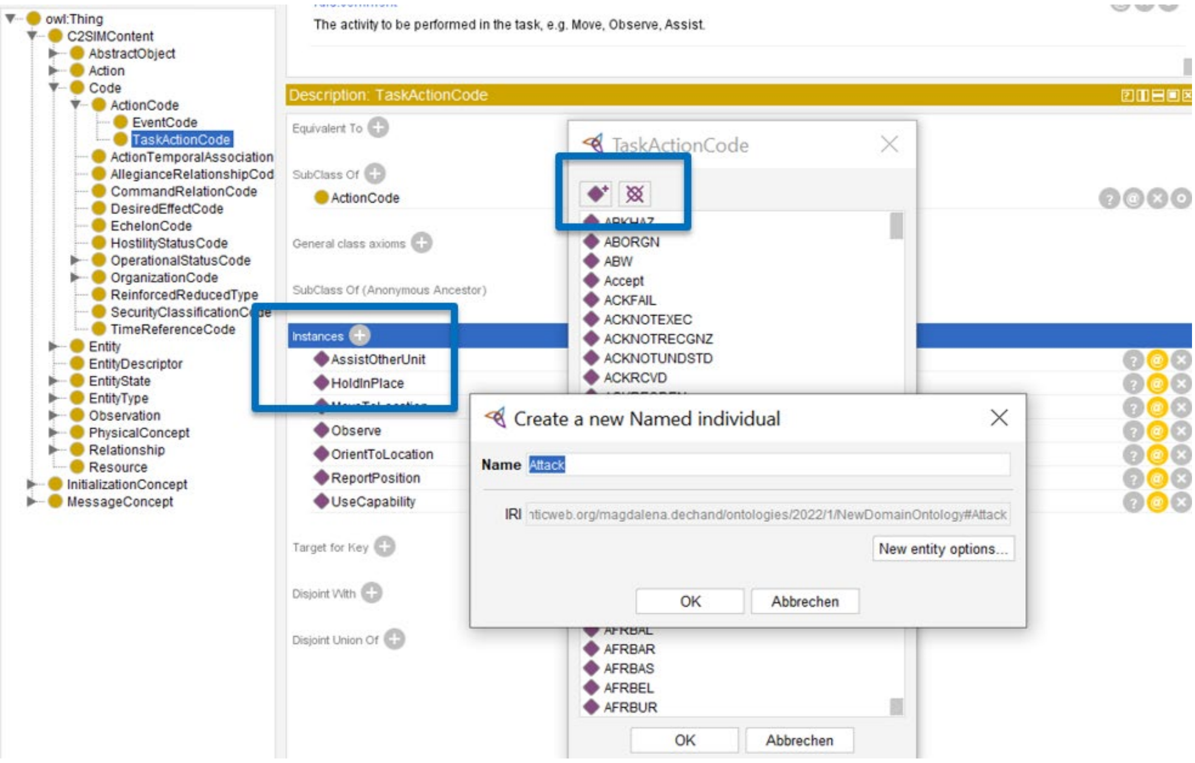
IRI a.dechand/ontologies/2022/1/NewDomainOntology#NewDomainReportConcent

New entity options...

OK | Abbrechen

Figure 22: Adding classes to taxonomy.

In some cases, it makes sense to add instances to a class with the help of an instance button to name and add a new instance or choose from the already existing ones (see Figure 23).



The activity to be performed in the task, e.g. Move, Observe, Assist.

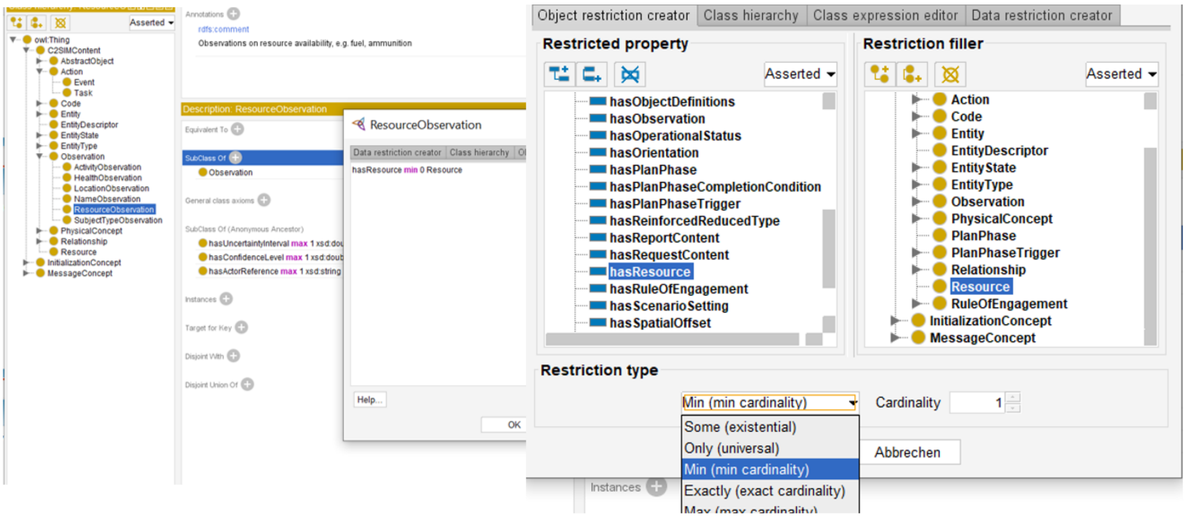
Create a new Named individual

Name: Attack

IRI: nticweb.org/magdalena.dechand/ontologies/2022/1/NewDomainOntology#Attack

Figure 23: Addition of instances.

A property restriction for a class can be created with the button SubclassOf that offers possibilities to generate an object property as plain text or with an interface providing all possible object properties to choose from and specify by *some*, *only*, *min*, *max* or *exactly* and the (class) range (see Figure 24).



Observations on resource availability, e.g. fuel, ammunition

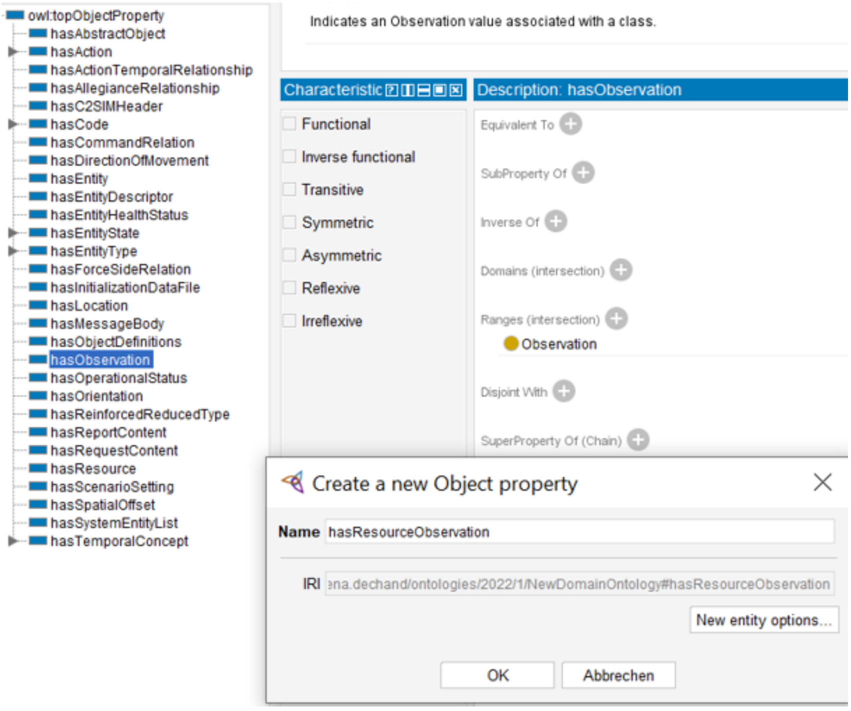
hasResource min 0 Resource

Restriction type: Min (min cardinality)

Cardinality: 1

Figure 24: Addition of object property restrictions.

If the existing property restrictions do not fit to one's purpose, new object properties can be created which are organized in a taxonomy. The range needs to be filled containing the class it refers to (see Figure 25). The other characteristics are not supported by the schema transformation tool [6].



Indicates an Observation value associated with a class.

Create a new Object property

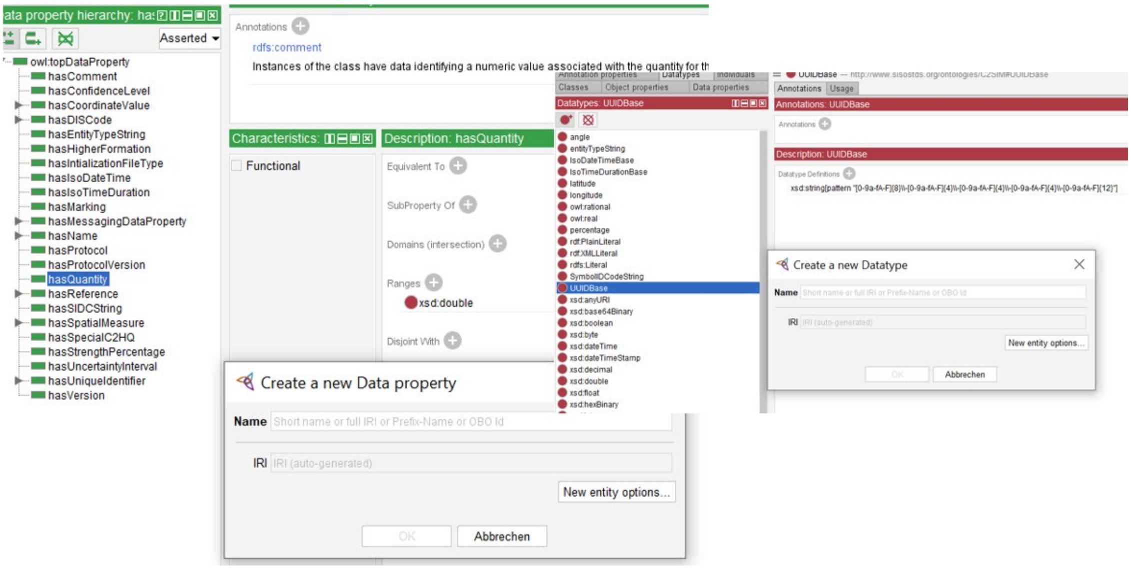
Name hasResourceObservation

IRI ena.dechand/ontologies/2022/1/NewDomainOntology#hasResourceObservation

Ranges (intersection): Observation

Figure 25: Create new object properties.

Regarding datatype property, a range represents a datatype like string, int or byte. Datatype properties are also organized in a taxonomy. Apart from predefined datatypes, new datatypes can be created using regular expressions (see Figure 26).



rdfs:comment

Instances of the class have data identifying a numeric value associated with the quantity for th

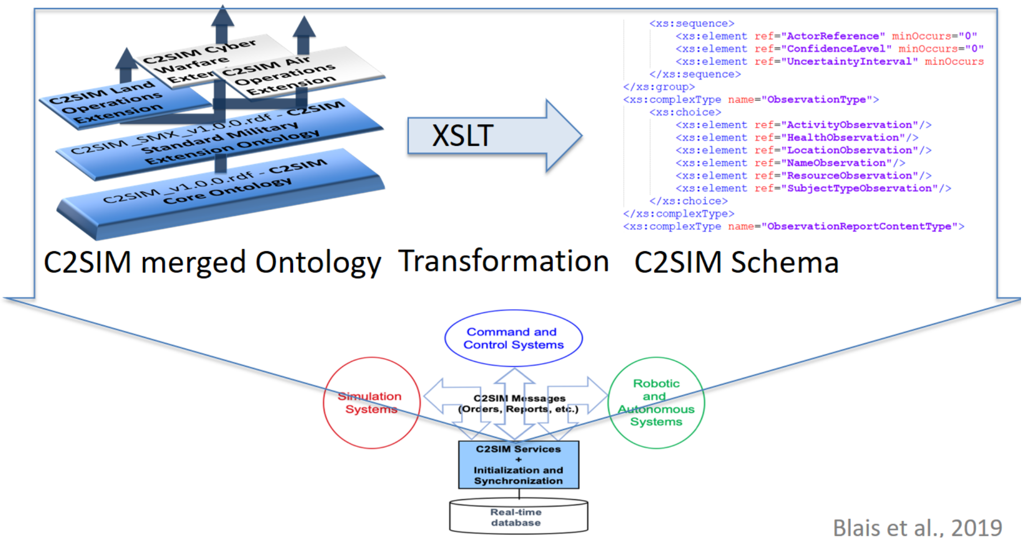
Ranges: xsd:double

UUIDBase — http://www.sisostds.org/ontologies/C2SIM#UUIDBase

Datatype Definitions: xsd:string[pattern "[0-9a-fA-F]{8}\\-[0-9a-fA-F]{4}\\-[0-9a-fA-F]{4}\\-[0-9a-fA-F]{4}\\-[0-9a-fA-F]{12}"]

Figure 26: Create new datatype properties.

Once a new ontology is modelled, further steps must follow to enable an information exchange between C2SIM systems. The ontologies of interest representing a logical data model need to be merged and transformed into a physical data model. The standard provides a transformation tool [7] to transform an XML schema from the C2SIM ontologies (see Figure 27).



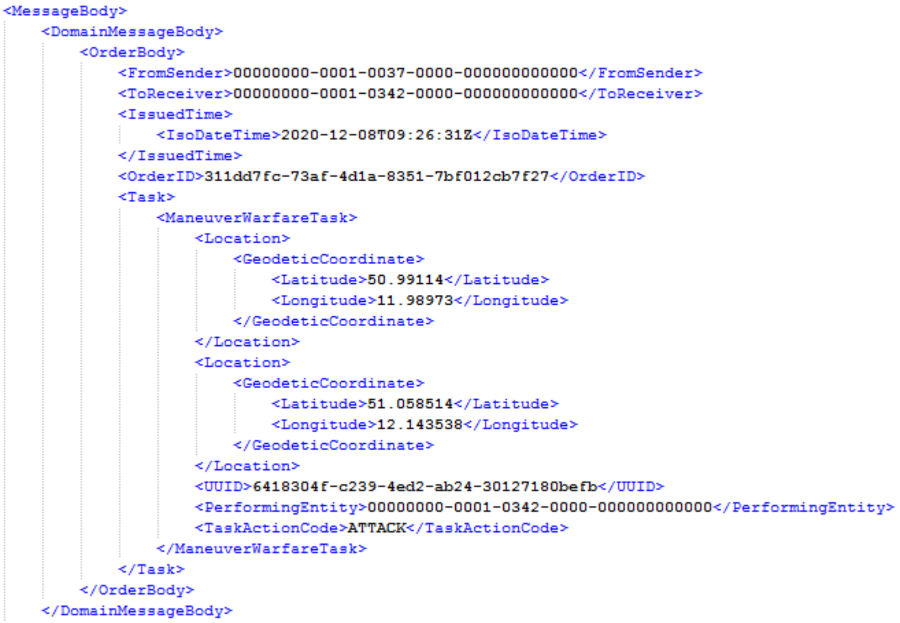
```
    <xs:sequence>
        <xs:element ref="ActorReference" minOccurs="0"
        <xs:element ref="ConfidenceLevel" minOccurs="0"
        <xs:element ref="UncertaintyInterval" minOccurs
    </xs:sequence>
</xs:group>
<xs:complexType name="ObservationType">
    <xs:choice>
        <xs:element ref="ActivityObservation"/>
        <xs:element ref="HealthObservation"/>
        <xs:element ref="LocationObservation"/>
        <xs:element ref="NameObservation"/>
        <xs:element ref="ResourceObservation"/>
        <xs:element ref="SubjectTypeObservation"/>
    </xs:choice>
</xs:complexType>
<xs:complexType name="ObservationReportContentType">
```

C2SIM Cyber Warfare Exten
C2SIM Air Operations Extension
C2SIM Land Operations Extension
C2SIM_SMX_v1.0.0.rdf - C2SIM Standard Military Extension Ontology
C2SIM_v1.0.0.rdf - C2SIM Core Ontology
XSLT
C2SIM merged Ontology
Transformation
C2SIM Schema
Command and Control Systems
Simulation Systems
C2SIM Messages (Orders, Reports, etc.)
Robotic and Autonomous Systems
C2SIM Services + Initialization and Synchronization
Real-time database

Blais et al., 2019

Figure 27: Procedure from ontology extension, transformation to schema to transfer data between systems.

The XML message in Figure 28 follows an automatically generated schema a *ManeuverWarefareTask* can look like. It uses information from different ontology layers like *DomainMessageBody* with its properties which in SMX. *ManeuverWarfareTask* is a LOX specific class which is linked to core concepts like *Location* or *TaskActionCode*.



```
<MessageBody>
    <DomainMessageBody>
        <OrderBody>
            <FromSender>00000000-0001-0037-0000-000000000000</FromSender>
            <ToReceiver>00000000-0001-0342-0000-000000000000</ToReceiver>
            <IssuedTime>
                <IsoDateTime>2020-12-08T09:26:31Z</IsoDateTime>
            </IssuedTime>
            <OrderID>311dd7fc-73af-4d1a-8351-7bf012cb7f27</OrderID>
            <Task>
                <ManeuverWarfareTask>
                    <Location>
                        <GeodeticCoordinate>
                            <Latitude>50.99114</Latitude>
                            <Longitude>11.98973</Longitude>
                        </GeodeticCoordinate>
                    </Location>
                    <Location>
                        <GeodeticCoordinate>
                            <Latitude>51.058514</Latitude>
                            <Longitude>12.143538</Longitude>
                        </GeodeticCoordinate>
                    </Location>
                    <UUID>6418304f-c239-4ed2-ab24-30127180befb</UUID>
                    <PerformingEntity>00000000-0001-0342-0000-000000000000</PerformingEntity>
                    <TaskActionCode>ATTACK</TaskActionCode>
                </ManeuverWarfareTask>
            </Task>
        </OrderBody>
    </DomainMessageBody>
```

Figure 28: XML message with values according to schema.

3.0 CONCLUSION

The C2SIM standard provides three ontologies as logical data model and a transformation process using the semantic model for a syntactical and physical representation of this information to exchange information for M&S. It also provides an easy way to extend the model according to own requirements such as for the use in different domains.

4.0 REFERENCES

[1] Biagini, M., & Corona, F. (2019). M&S-Based Robot Swarms Prototype. In J. Mazal, Modelling and Simulation for Autonomous Systems. MESAS 2018. Lecture Notes in Computer Science, Vol 11472 (pp. 285-301). Cham, Schweiz: Springer.

[2] Biagini, M., Corona, F., Wolski, M., & Schade, U. (2017). Conceptual Scenario Supporting Extension of C2SIM to Autonomous Systems. 22nd ICCRTS. Los Angeles, CA: CCRP.

[3] Blais, C. (2022a): An Update on Efforts to Extend the Command and Control Systems – Simulation Systems Interoperation (C2SIM) Standard for Exchanging Cybersecurity Information. 2022 Virtual Simulation Innovation Workshop (SIW).

[4] Blais, C. (2022b): Reconciling the Command and Control Systems – Simulation Systems Interoperation (C2SIM) Standard with the NATO Education and Training Network (NETN) Federation Object Model. 2022 Virtual Simulation Innovation Workshop (SIW).

[5] Blais, C., Dechand, M., Dembach, M. & Singapogu, S. (2021): The Use of Reasoning with Command and Control System to Simulation System Interoperation (C2SIM) Standard. 2021 Virtual Simulation Innovation Workshop (SIW).

[6] Blais, C., Gautreau, B., Schade, U., Sikorski, L., Wolski, M., & Singapogu, S. (2019). Transformation Process for Generating an Extensible Markup Language (XML) Schema from a Formal Ontology for Practical Application in C2SIM Implementations. 2019 Winter Simulation Innovation Workshop. Orlando, FL: SISO.

[7] Blais, C., Reece, D., & Singapogu, S. (2019). From Information Description to Information Understanding: The Role of Ontology in Emerging SISO Standards. 2019 Winter Simulation Innovation Workshop. Orlando, FL: SISO

[8] Dechand, M. (2022): C2SIM Ontologies: MSG-194 Technical Course on Employing the C2-Simulation Interoperation (C2SIM) Standard for Coalition Military Operations and Exercises.

[9] Dechand, M., Sikorski, L., Trautwein, I., Gautreau, B., Bouvier, E., & Khimeche, L. (2019). Development of an Air Operation eXtension with the (future) C2SIM standard. NATO Modelling and Simulation Group Symposium. Wien.

[10] Dembach, M., Blais, C., Dechand, M. (2022): A Java-based application for the generation of XML-schemas from C2SIM ontologies. 2022 Virtual Simulation Innovation Workshop (SIW).

[11] Pullen, J. M., & Khimeche, L. (2014). Advances in Systems and Technologies toward Interoperating Operational Military C2 and Simulation Systems. 19th International Command and Control Research and Technology Symposium (ICCRTS). Alexandria, VA: CCRP.

[12] Pullen, J. M., Corner, D., & Wittman, R. (2013). Next Steps in MSDL and C-BML Alignment for Convergence. IEEE 2013 Spring Simulation Interoperability Workshop. San Diego, CA: SISO.

[13] Pullen, J. M., Corner, D., Blais, C., Reece, D., Ruth, J., & Singapogu, S. (2019). Command and Control System to Simulation System Interoperation: Development of the C2SIM Standard. Winter Simulation Innovation Workshop. Orlando, FL: SISO.

[14] Protégé Ontology Tool: https://protege.stanford.edu/

[15] C2SIM Products: https://www.sisostds.org/Default.aspx?tabid=105&EntryId=51847

[16] MIP: Welcome – Welcome to the MIM –p MIP Information Model (mimworld.org).