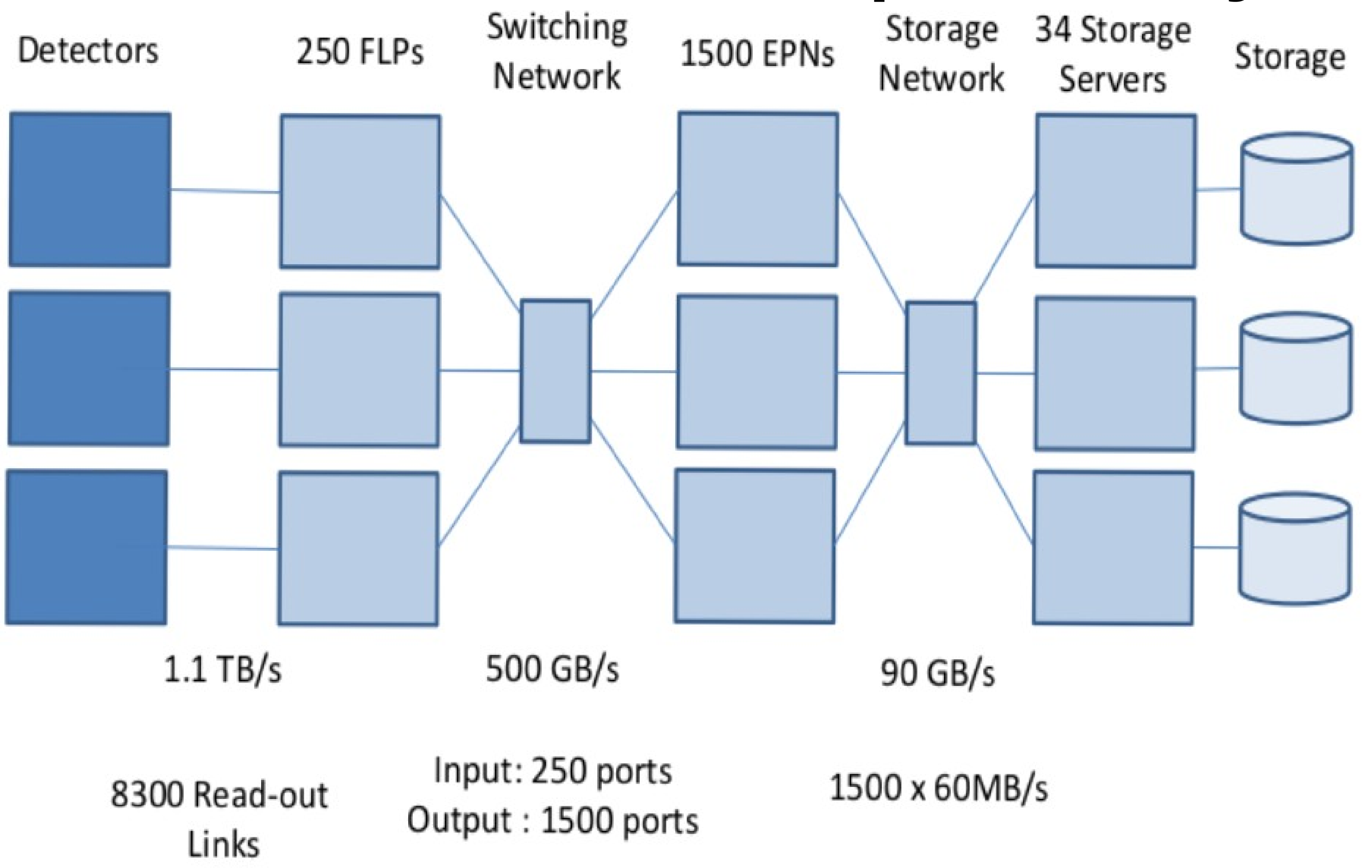
DDS Topology Editor

Aleksandar Rusinov
August 28th, 2015

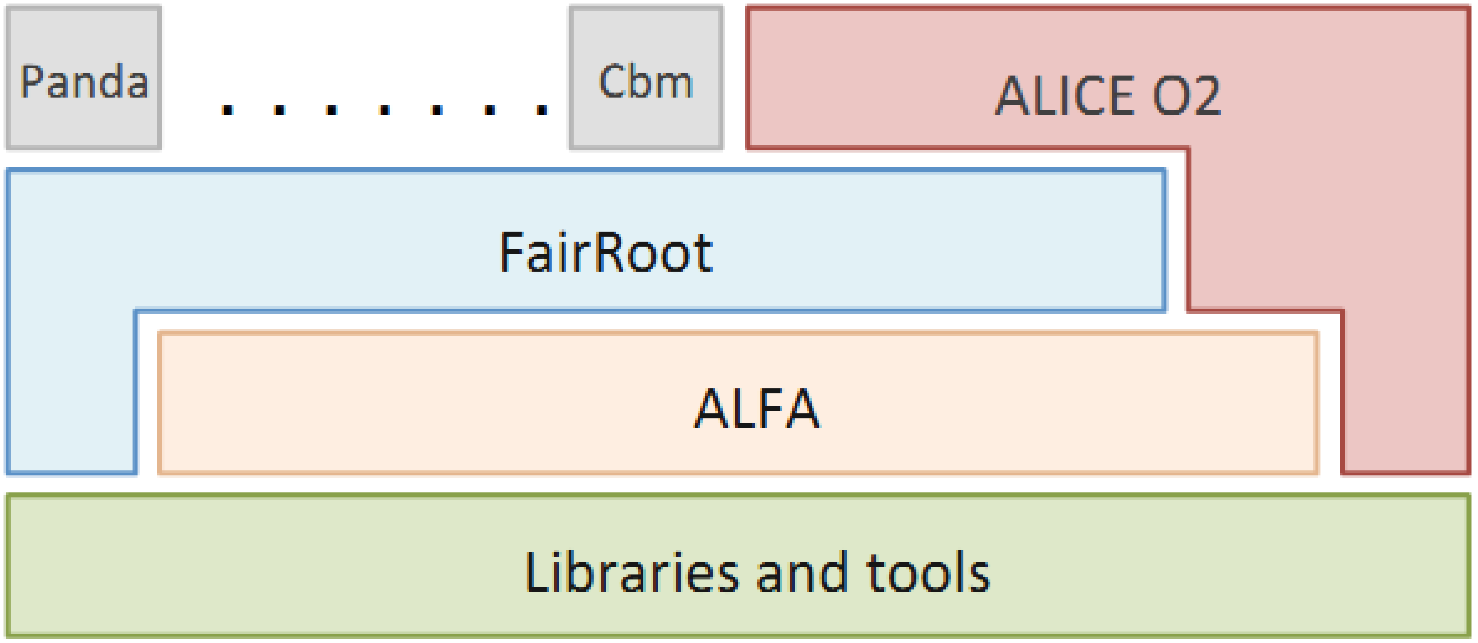
Outline

- What is Alice O2 and DDS
- Objectives for the DDS Topology Editor
- Method developed
- Conclusion and next steps

The global hardware architecture of the ALICE O2 computer system



O2 software ecosystem



The Dynamic Deployment System

Is a tool-set that automates and simplifies the process of deployment of user defined processes with their dependencies on any resource management system using a given topology

Basic concepts

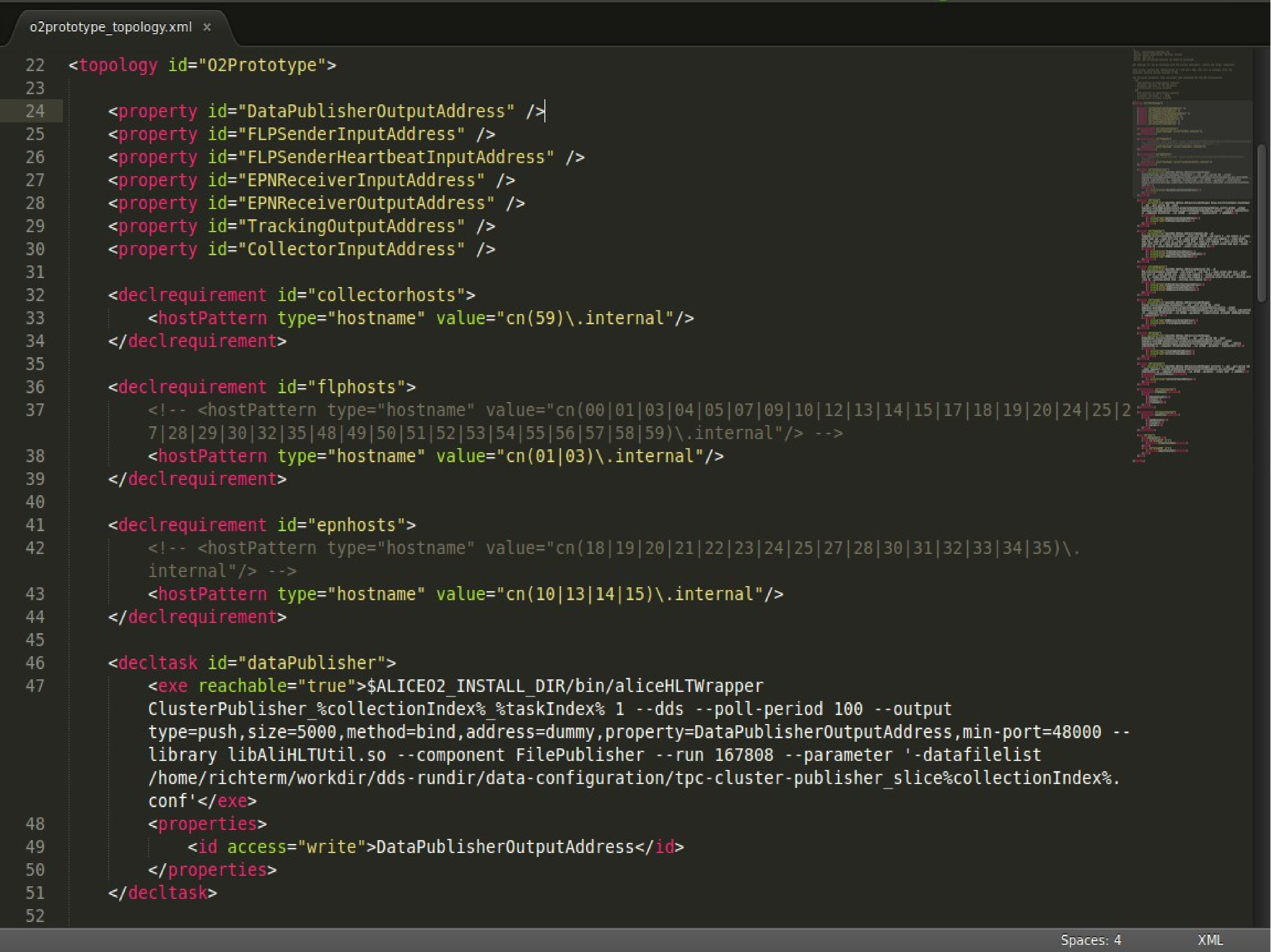
- Treats users tasks as a black boxes
- Does not depend on RMS(provides deployment via SSH, when no RMS is present)
- Provides a rules based execution of tasks

The Dynamic Deployment System

- The system takes a topology description file (XML) as it's input.
- Users describe desired tasks and dependencies by modifying the xml file.

DDS Topology Description XSD

```
<topology id="myTopology">
 [... Definition of tasks, properties, and collections ...]
 <main name="main">
 [... Definition of the topology itself, where groups are defined ...]
 </main>
</topology>
```



```xml
<topology id="O2Prototype">

    <property id="DataPublisherOutputAddress" />
    <property id="FLPSenderInputAddress" />
    <property id="FLPSenderHeartbeatInputAddress" />
    <property id="EPNReceiverInputAddress" />
    <property id="EPNReceiverOutputAddress" />
    <property id="TrackingOutputAddress" />
    <property id="CollectorInputAddress" />

    <declrequirement id="collectorhosts">
        <hostPattern type="hostname" value="cn(59)\.internal"/>
    </declrequirement>

    <declrequirement id="flphosts">
        <!-- <hostPattern type="hostname" value="cn(00|01|03|04|05|07|09|10|12|13|14|15|17|18|19|20|24|25|27|28|29|30|32|35|48|49|50|51|52|53|54|55|56|57|58|59)\.internal"/> -->
        <hostPattern type="hostname" value="cn(01|03)\.internal"/>
    </declrequirement>

    <declrequirement id="epnhosts">
        <!-- <hostPattern type="hostname" value="cn(18|19|20|21|22|23|24|25|27|28|30|31|32|33|34|35)\.internal"/> -->
        <hostPattern type="hostname" value="cn(10|13|14|15)\.internal"/>
    </declrequirement>

    <decltask id="dataPublisher">
        <exe reachable="true">$ALICEO2_INSTALL_DIR/bin/aliceHLTWrapper ClusterPublisher_%collectionIndex%_%taskIndex% 1 --dds --poll-period 100 --output type=push,size=5000,method=bind,address=dummy,property=DataPublisherOutputAddress,min-port=48000 --library libAliHLTUtil.so --component FilePublisher --run 167808 --parameter '-datafilelist /home/richterm/workdir/dds-rundir/data-configuration/tpc-cluster-publisher_slice%collectionIndex%.conf'</exe>
        <properties>
            <id access="write">DataPublisherOutputAddress</id>
        </properties>
    </decltask>

```

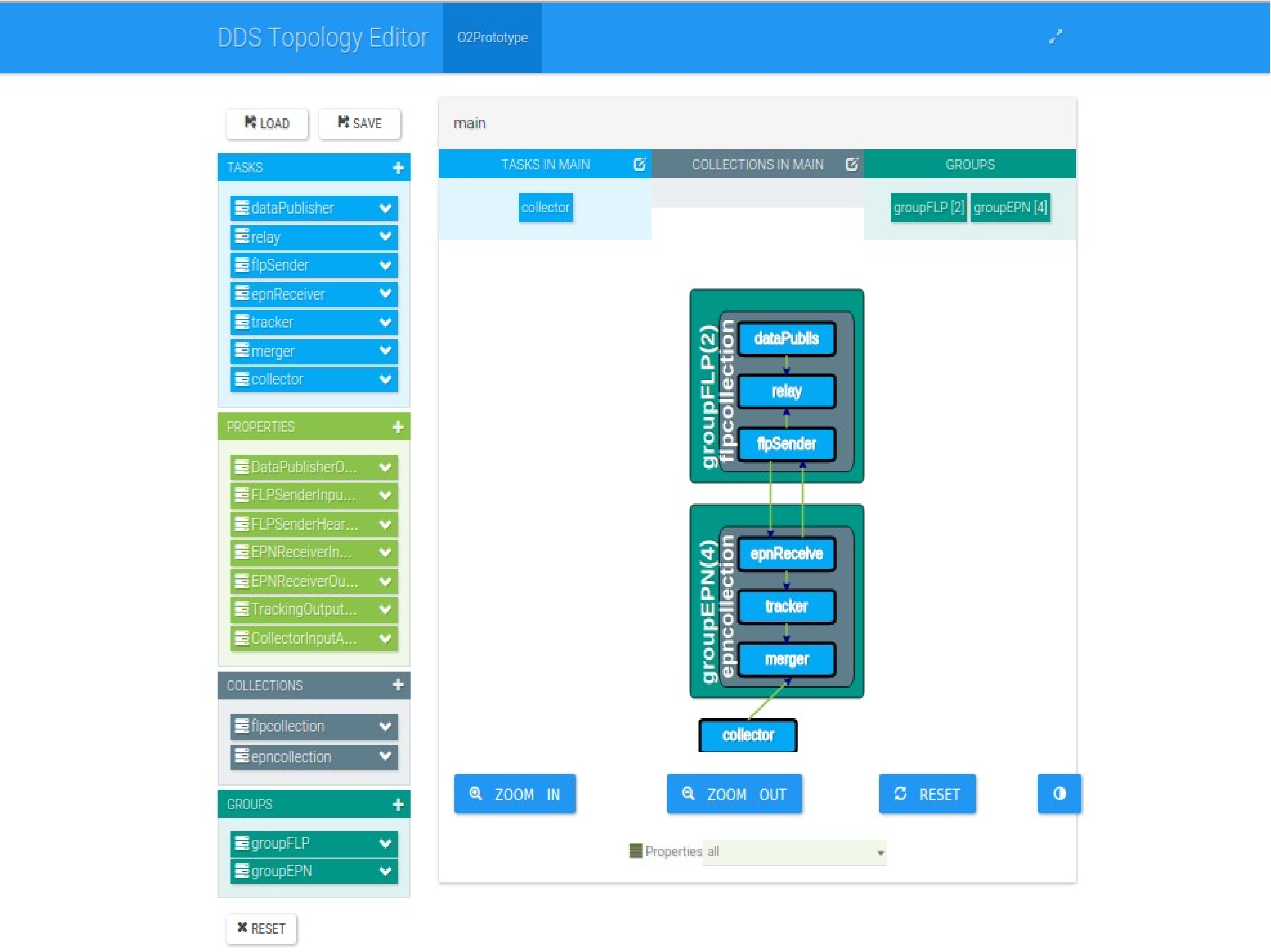
DDS Topology Editor Main Objective

- To provide the DDS user with an intuitive platform for creating the desired topology.

Solution

An interactive web tool that allows: creation, modification and visualization of a DDS topology.

- ✓ WEB based GUI
- ✓ Client-based platform
- ✓ Responsive uniform design and visualization
- ✓ Capability to save User’s descriptions in XML file as an input to DDS or to be reloaded later

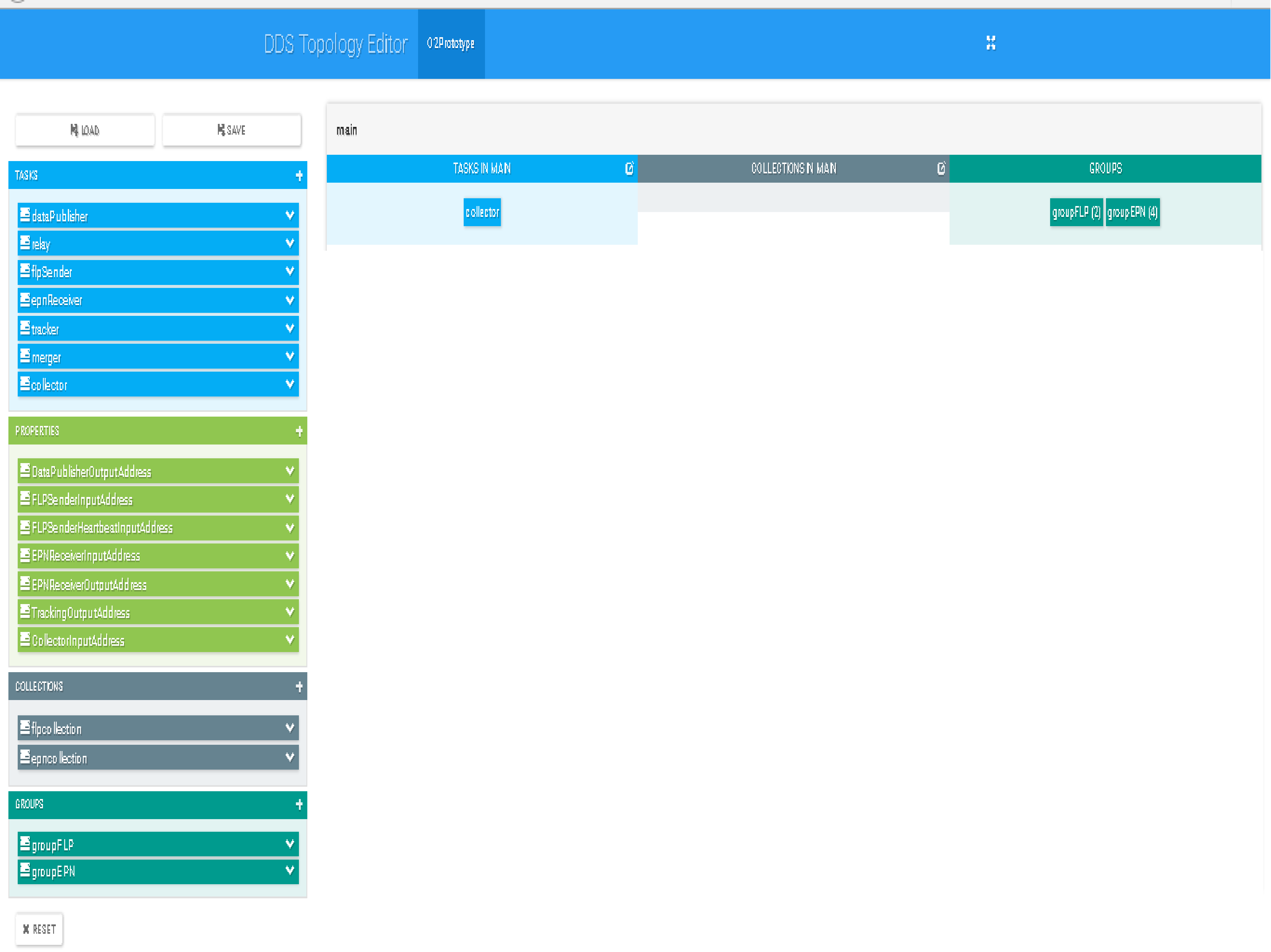

DDS Topology Editor
O2Prototype
LOAD
SAVE
TASKS
dataPublisher
relay
flpSender
epnReceiver
tracker
merger
collector
PROPERTIES
DataPublisherO...
FLPSenderInpu...
FLPSenderHear...
EPNReceiverIn...
EPNReceiverOu...
TrackingOutput...
CollectorInputA...
COLLECTIONS
flpcollection
epncollection
GROUPS
groupFLP
groupEPN
RESET
main
TASKS IN MAIN
COLLECTIONS IN MAIN
GROUPS
collector
groupFLP [2]
groupEPN [4]
groupFLP(2)
flpcollection
dataPublis
relay
flpSender
groupEPN(4)
epncollection
epnReceive
tracker
merger
collector
ZOOM IN
ZOOM OUT
RESET
Properties
all

DDS topology editor graphical input form

Graphical forms input corresponding to two sections of XML file –

- declarative part
- executable part (main),

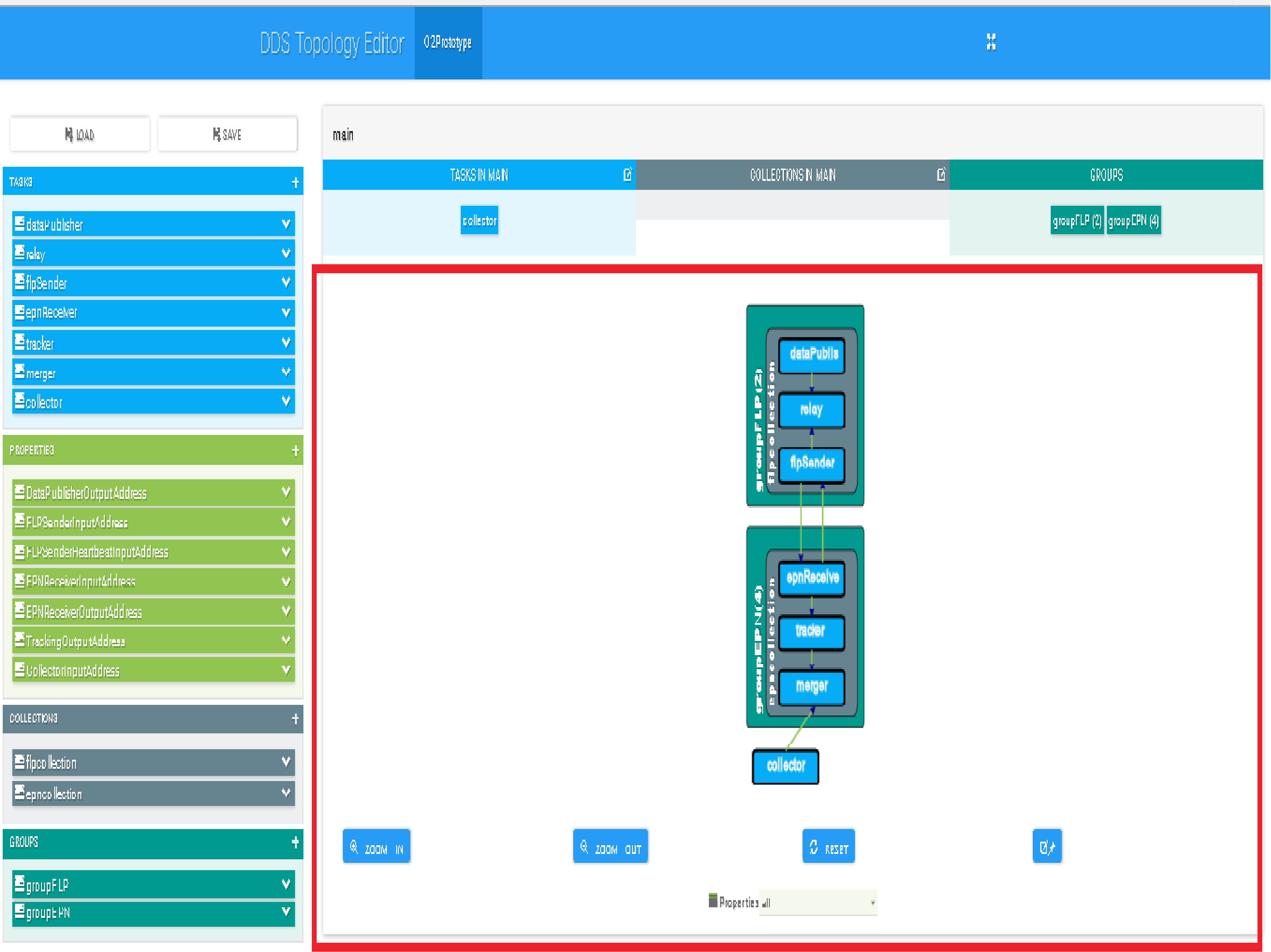
according to XSD definition of topology data elements


DDS Topology Editor
O2Prototype
LOAD
SAVE
TASKS
dataPublisher
relay
flpSender
epnReceiver
tracker
merger
collector
PROPERTIES
DataPublisherOutputAddress
FLPSenderInputAddress
FLPSenderHeartbeatInputAddress
EPNReceiverInputAddress
EPNReceiverOutputAddress
TrackingOutputAddress
CollectorInputAddress
COLLECTIONS
flpcollection
epncollection
GROUPS
groupFLP
groupEPN
RESET
main
TASKS IN MAIN
COLLECTIONS IN MAIN
GROUPS
collector
groupFLP (2)
groupEPN (4)

DDS topology editor graph visualisation

Interactive graph representation of executable (main):

- Tasks
- Collections,
- Groups,
- Properties (edges)


DDS Topology Editor
O2Prototype
LOAD
SAVE
main
TASKS IN MAIN
COLLECTIONS IN MAIN
GROUPS
collector
groupFLP (2)
groupEPN (4)
TASKS
dataPublisher
relay
flpSender
epnReceiver
tracker
merger
collector
PROPERTIES
DataPublisherOutputAddress
FLPSenderInputAddress
FLPSenderHeartbeatInputAddress
EPNReceiverInputAddress
EPNReceiverOutputAddress
TrackingOutputAddress
CollectorInputAddress
COLLECTIONS
flpcollection
epncollection
GROUPS
groupFLP
groupEPN
groupFLP(2)
flpcollection
dataPublis
relay
flpSender
groupEPN(4)
epncollection
epnReceive
tracker
merger
collector
ZOOM IN
ZOOM OUT
RESET
Properties all

Method

- Developed using JavaScript/HTML/CSS packet
- With extensive use of open source JavaScript Libraries.
  - ReactJS for the general structure and component.
  - JointJS for the graph visualisation.

The Editor Allows

- Viewing of an existing topology
- Editing of a topology
- Creation of a new topology
- Saving of topology description to DDS XML file

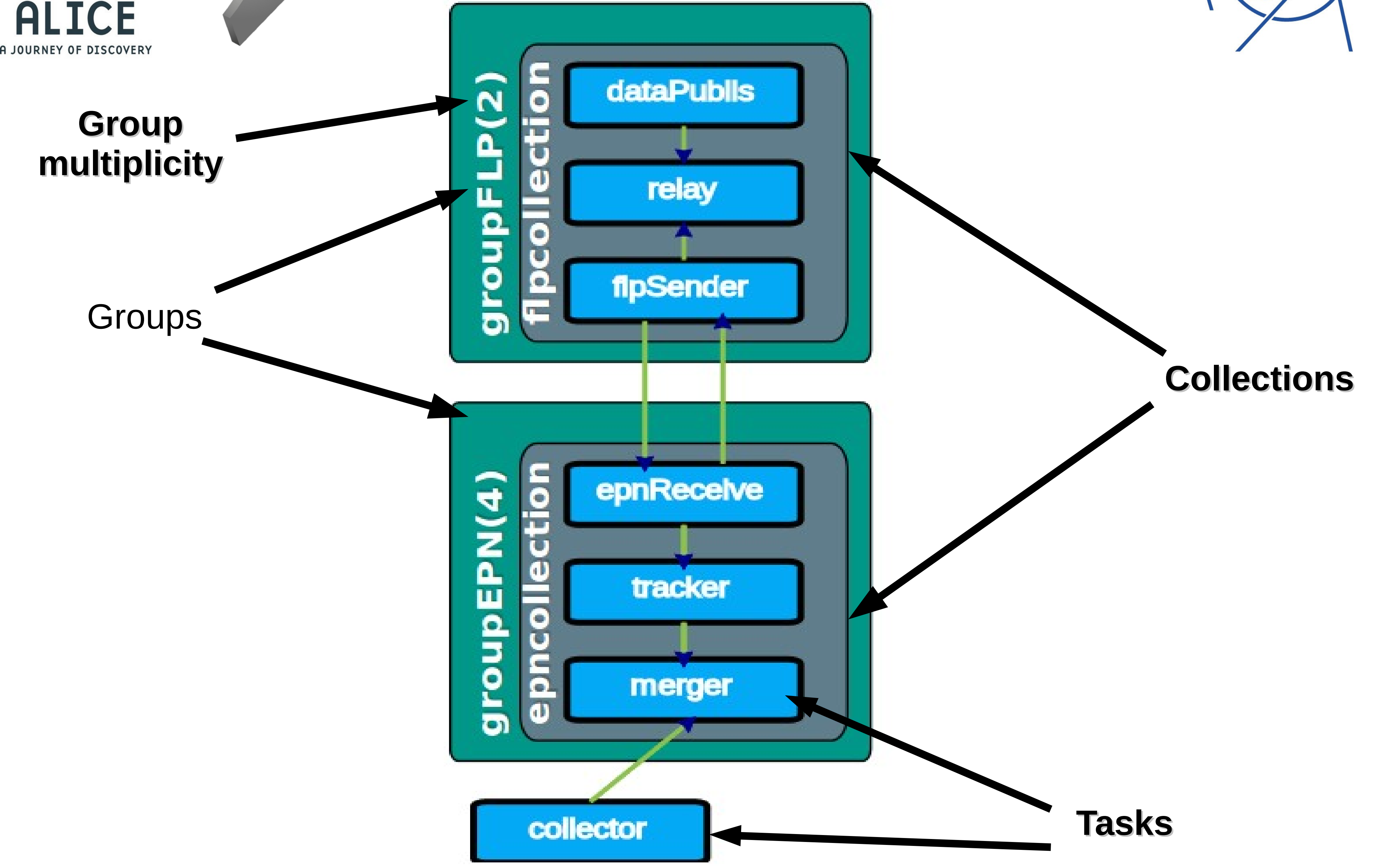
The Editor has

- An object model representing the DDS structure
- Editing Menu for the declarative part of the DDS
- Editing Menu for the Main part of the DDS
- Graph visualisation of the topology
- Menu with visualisation settings

Visualisation

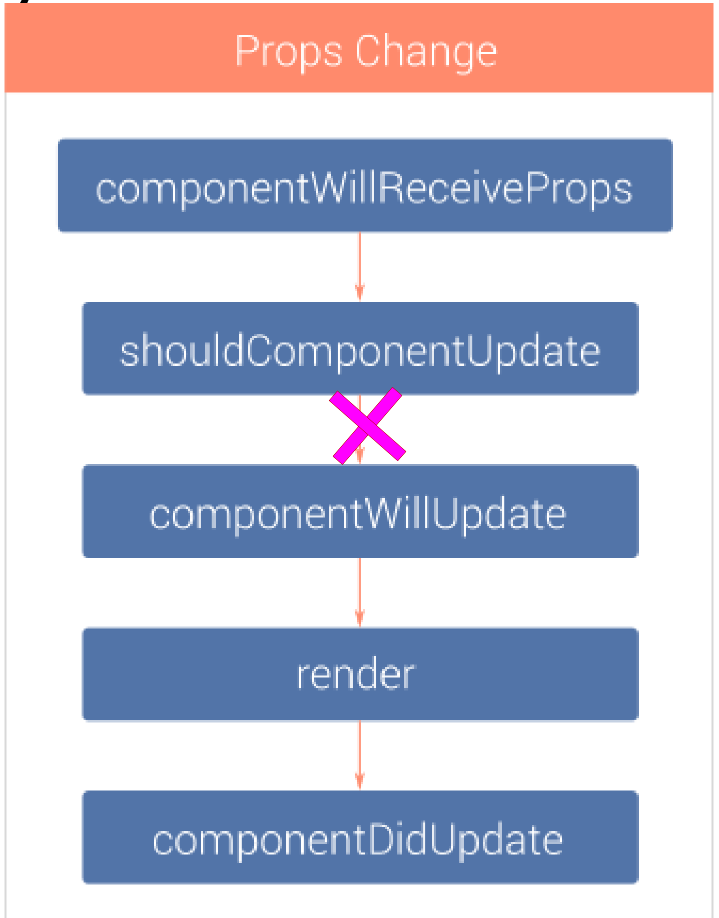
- Graph visualisation of the topology build with JointJS:
- Hierarchical diagrams of the collection and group nesting combined with a graph structure on a task level.
- Each property is represented as a directed edge between the tasks where:
  - The write task is a parent
  - The read task is a child

Visualisation



Challenges

- No suitable native to ReactJS method to Integrate the visualization into a React component to transfer the topology data
  - Made use of a Black Box design pattern in ReactJS and break the rendering life-cycle while still generating the svg visualisation canvas of the topology
  - The graph generation takes place in shouldComponentUpdate



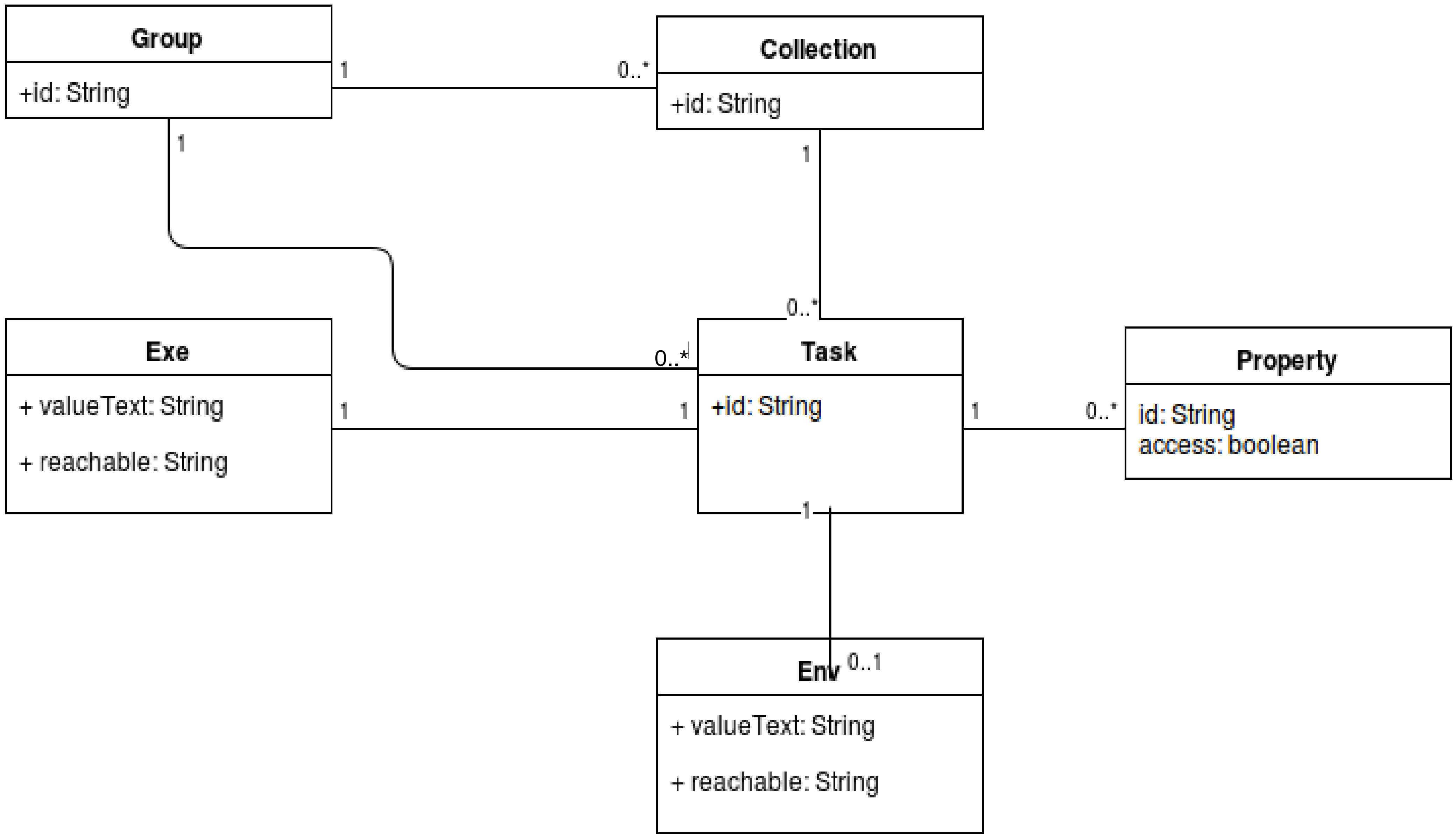
ALICE
A JOURNEY OF DISCOVERY

CERN

Challenges

- The native graph generation of JointJS is pretty time consuming for a large number of edges – each line(graph edge) creates three source files
  - Used a lightweight library vectorizer.js that represents a link as a simple svg line and written methods for a dynamic change on drag and drop.

Structure


Group
+id: String
Collection
+id: String
Exe
+ valueText: String
+ reachable: String
Task
+id: String
Property
id: String
access: boolean
Env
+ valueText: String
+ reachable: String
1
0..*
1
1
0..*
0..*
1
1
1
0..*
1
0..1

ALICE
A JOURNEY OF DISCOVERY

CERN

Demonstration

Conclusion

- The main objective of the project is achieved – a good level of visualization for the DDS topology present in the Current topology editor.
- Smooth integration of the graph with the editor part
- The graph part of the topology editor has a set of control actions – zoom, pann, interactive mode and property selection.

Next steps

- Investigate ways to show accurately the direction of the process flow described in the topology file (we can not infer that from the DDS). For example:
  - Put an indication by additional attribute in the XML
  - Make a property that creates a DAG representing the process flows
- Consider how the visualisation will look like in a more complicated topology and make the necessary corrections
- Enable editing in the visual graph – adding a link between two tasks for a new property

Thank you!

Questions?

Topology Editor	http://alex92rus.github.io/DDS-topology-editor/

DDS	http://dds.gsi.de/