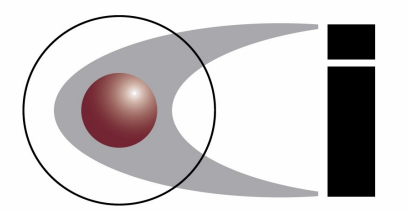
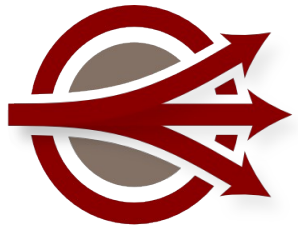
OpenDDS Modeling Toolkit

Capturing Middleware using UML Models



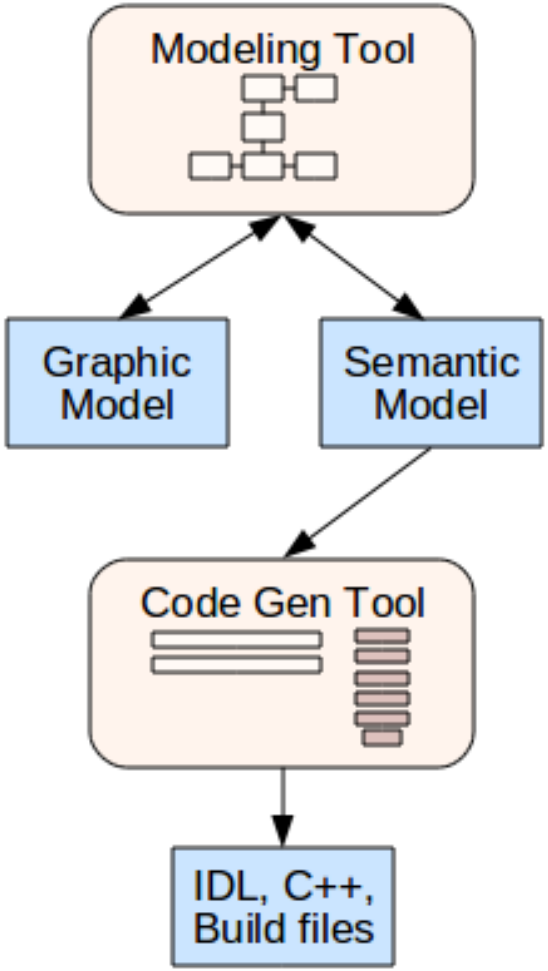
www.ociweb.com



www.opendds.org

Overview

- Eclipse based model capture
  - Middleware
  - Data
  - Quality of Service Policies
  - Code generation
  - Validation
- Files and References
  - XMI – tool specific, graphical information
  - XML – DDS semantics, Deployment information
- Application Integration
  - Build Process
  - Support Libraries
  - Model and Application Organization



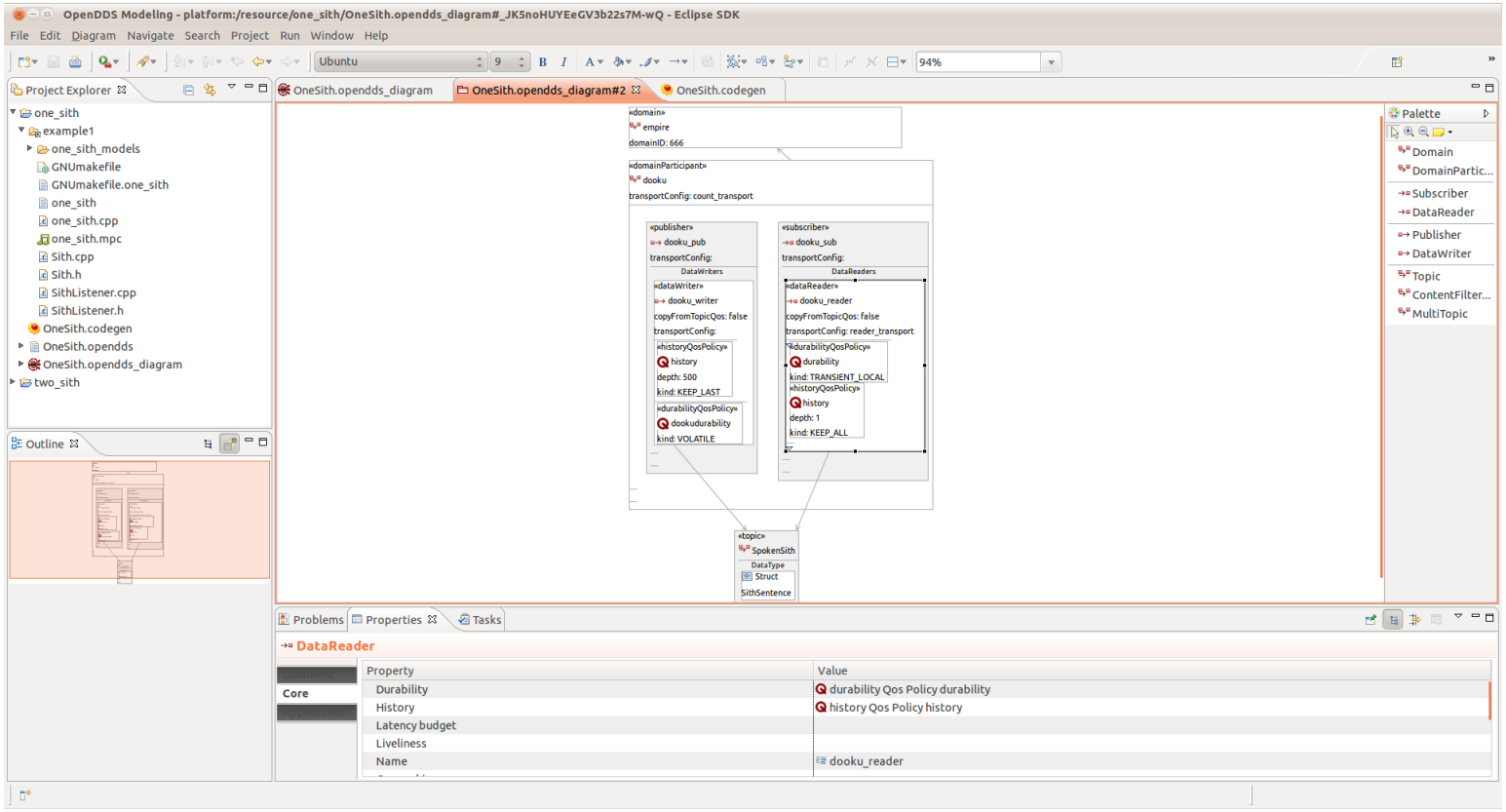
Motivation

- SBIR N08-116
  - "Open Data Distribution Service (DDS) for use in a real time simulation laboratory environment"
  - NAVAIR E2C System Test and Evaluation Laboratory (ESTEL)
  - Wanted FOSS solution to replace HLA and DIS simulation transports
    - OpenDDS, extended to full standards compliance
  - Wanted modeling toolkit to obviate need for middleware expertise of their domain programmers
    - Eclipse provided FOSS solution
- Modeling advantages
  - Platform independence
    - Focus on functionality and behavior
  - Technology migration
    - Move forward in time
    - Move to different implementations
  - Middleware as an abstraction

We Use The Toolkit

- Small scale project
  - Migrated JSON data to IDL
  - Load balanced jobs using ContentFilteredTopics
  - Single model usage
- Medium scale project
  - Defined complex topology using PARTITION and ContentFilteredTopics
  - USERDATA and BuiltinTopics used for job scheduling
  - Shared model usage
- Lessons learned were incorporated into the toolkit

Toolkit



Eclipse Based Toolkit

- Eclipse IDE
  - Open source, freely available, ubiquitous
  - Basis for model capture, file organization, code generation
- Eclipse Modeling Framework
  - Define model semantics
  - Generate model editing support
- Graphical Modeling Project
  - Provides standard graphical editors, familiar to Eclipse users
  - Bind semantic model to graphical elements
- Eclipse Plug-ins
  - Organize features and provide standard distribution mechanism
    - http://www.opendds.org/modeling/eclipse
    - http://www.opendds.org/modeling/eclipse/opendds_modeling_site.zip
  - Additional forms editors for deployment and customization
  - User interface for code generation

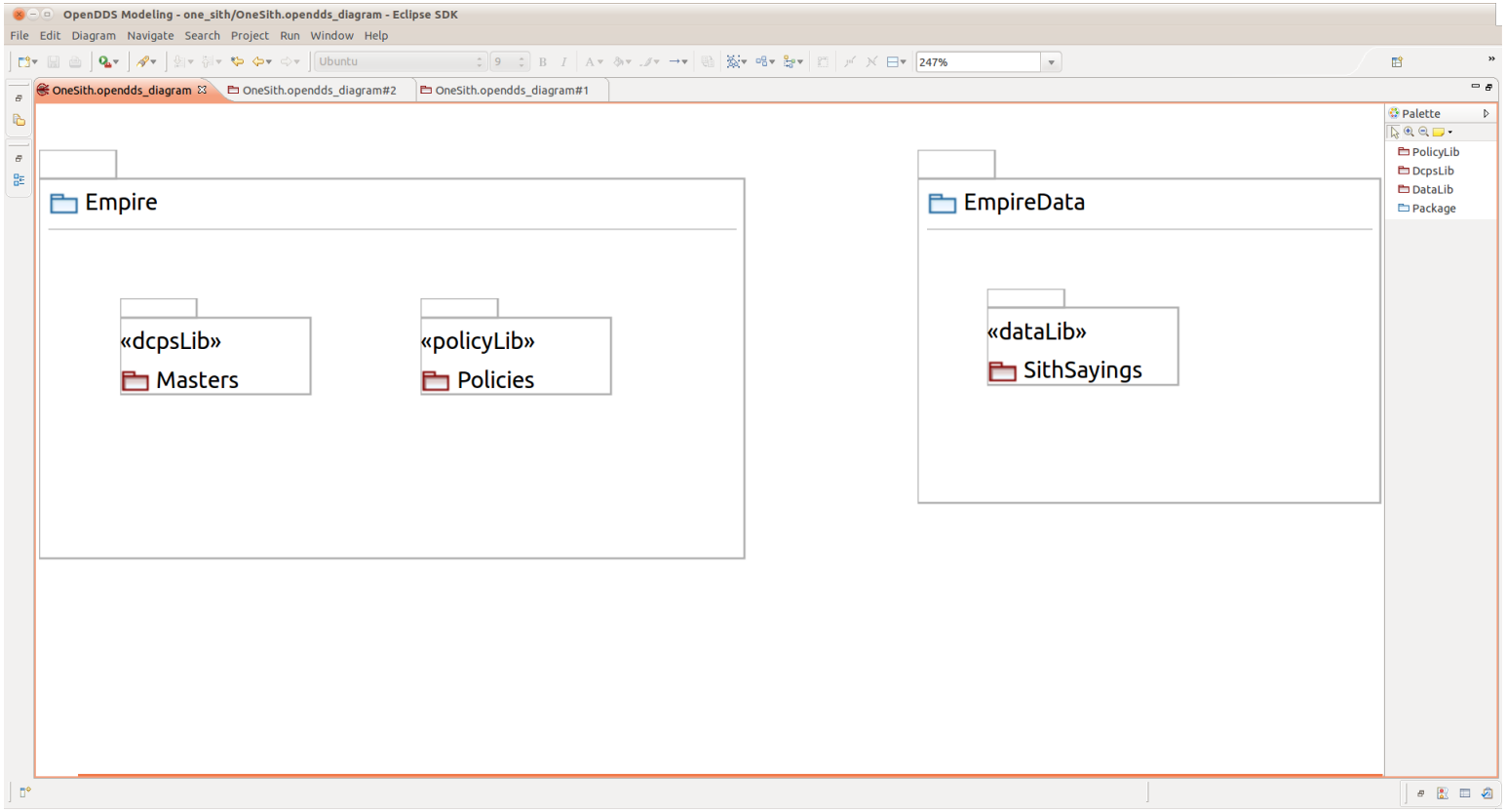
Editors

- Graphical model capture
  - Package diagram
  - Middleware diagram
  - QoS policy diagram
  - Data definition diagram
- Forms based editors
  - Model customizations
  - Build support information
- Code generation
  - Define source model and target directory for results

Meta-models

- Use Eclipse Ecore syntax
  - Similar to but not eMOF
  - Matched to (required by) EMF
- Originally based on ptc/10-05-15
  - Unable to map completely to Ecore syntax
  - Included application binding in the middleware semantics
- Several separate packages
  - Core, Domain, Topic, Types, DCPS, QoS, Enumerations
  - Generator package added to specify toolkit specific deployment information (application binding)
- Mapped onto graphical editors
  - Core, Domain, Topic, DCPS ==> DCPS editor
  - QoS ==> QoS editor
  - Types ==> Data editor
  - Generator ==> Customization and build support forms

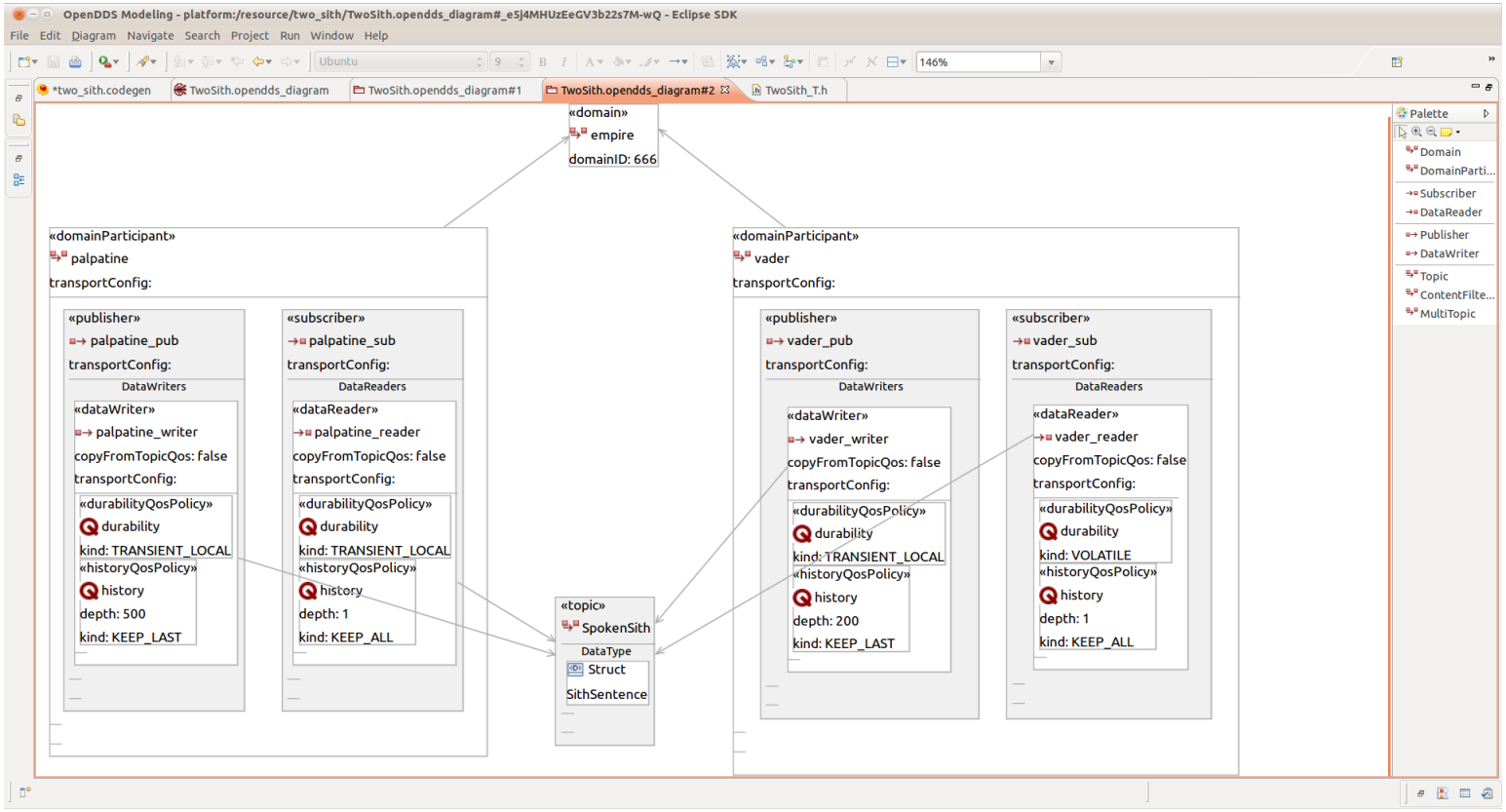
Package Diagram



Package Editor

- Top level diagram
- Entry point for all models
- Includes structural packages
  - 'modules' in generated IDL files
  - 'namespaces' in generated C++ files
- Organizes model
  - use of locally defined 'libraries'
  - use of externally referenced 'libraries'

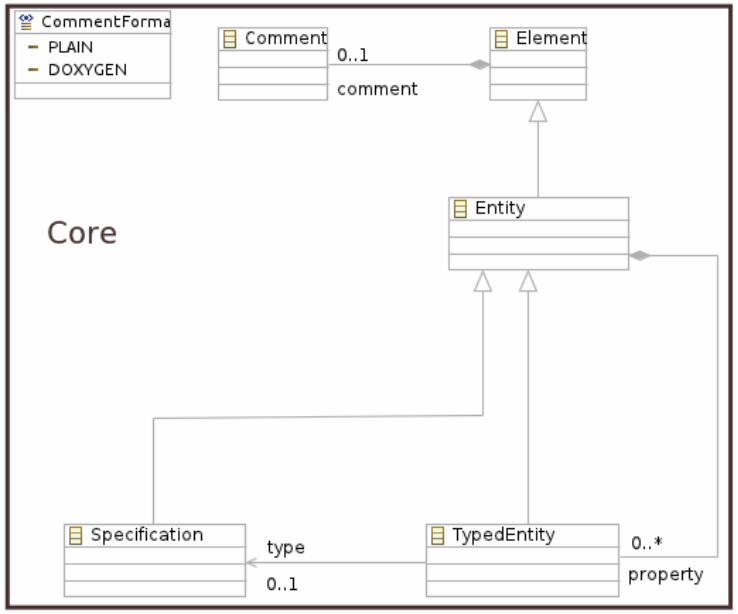
Middleware Diagram

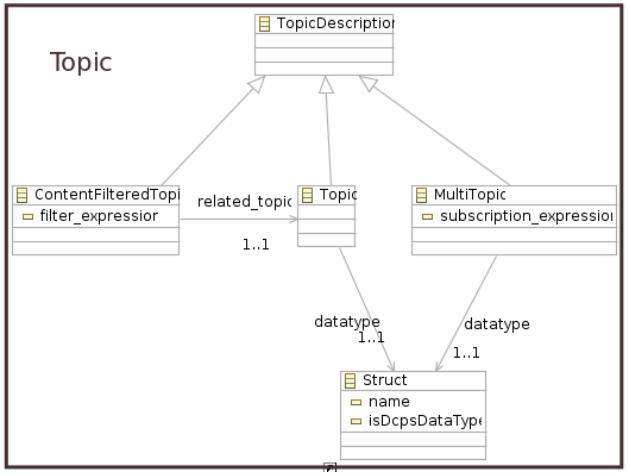


Middleware Editor

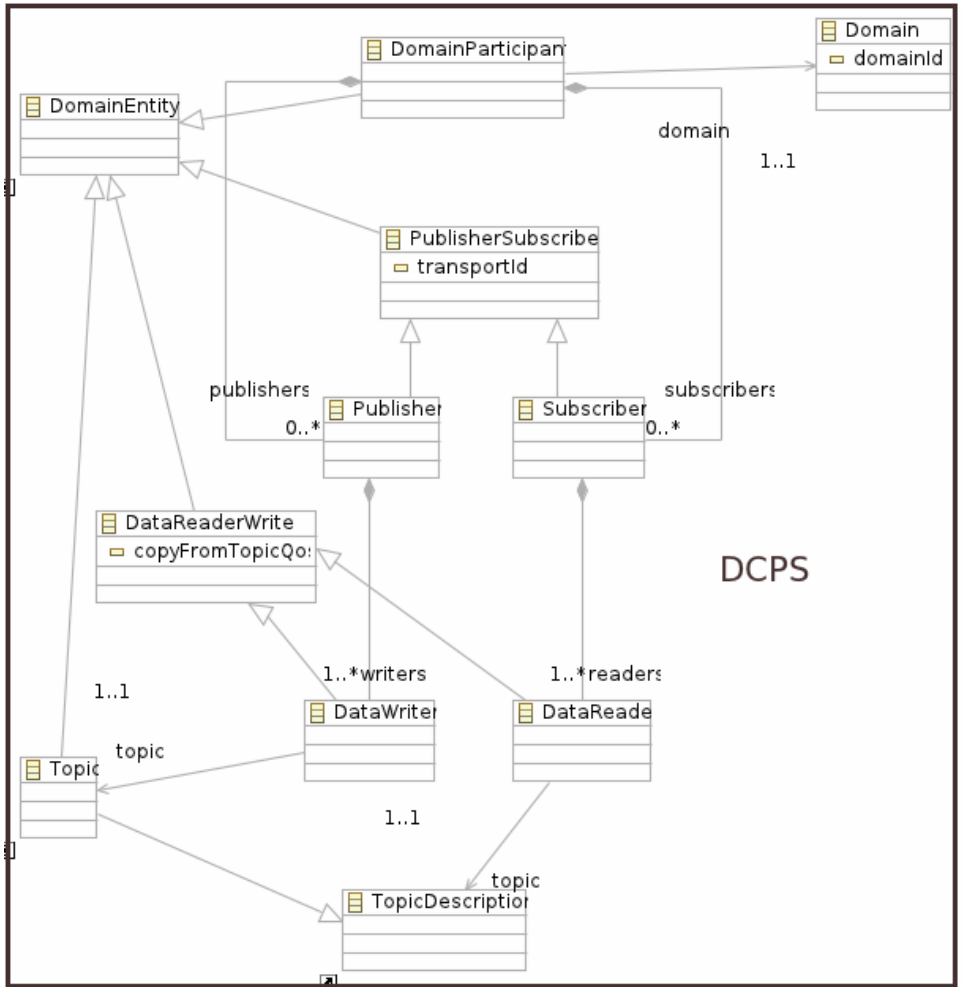
- Includes DDS semantic elements
  - Domain, DomainParticipant, Subscriber, Publisher, DataWriter, DataReader, *Topic
- Includes relationships between these elements
  - Associations
    - Bind participants to domains
    - Bind readers and writers to topics
  - Containment
    - Participants contain publishers and subscribers
    - Publishers contain writers, Subscribers contain readers
    - All contain QoS policies
- Attribute definitions
  - Semantic attributes
    - DomainID
  - Model attributes
    - Element names

Semantic Content

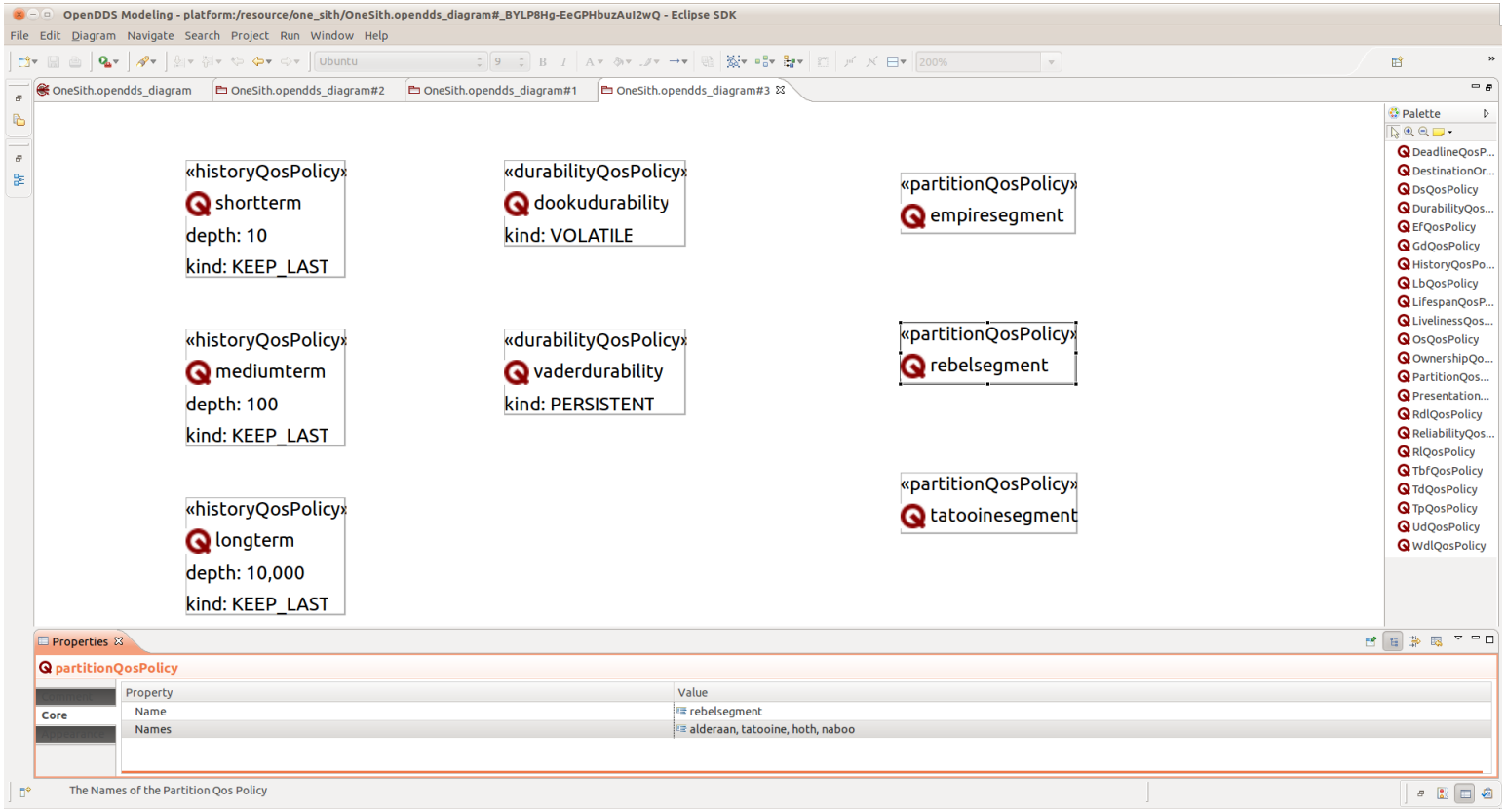




Semantic Content – cont.



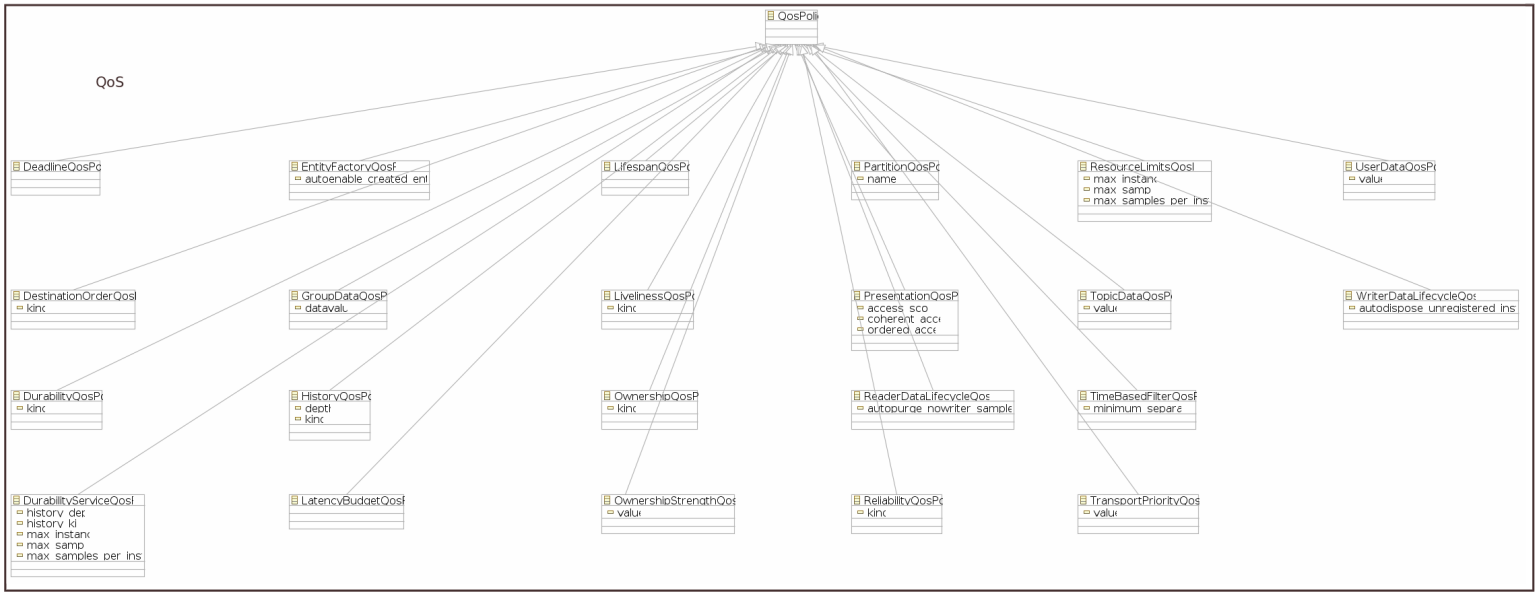
QoS Policy Diagram



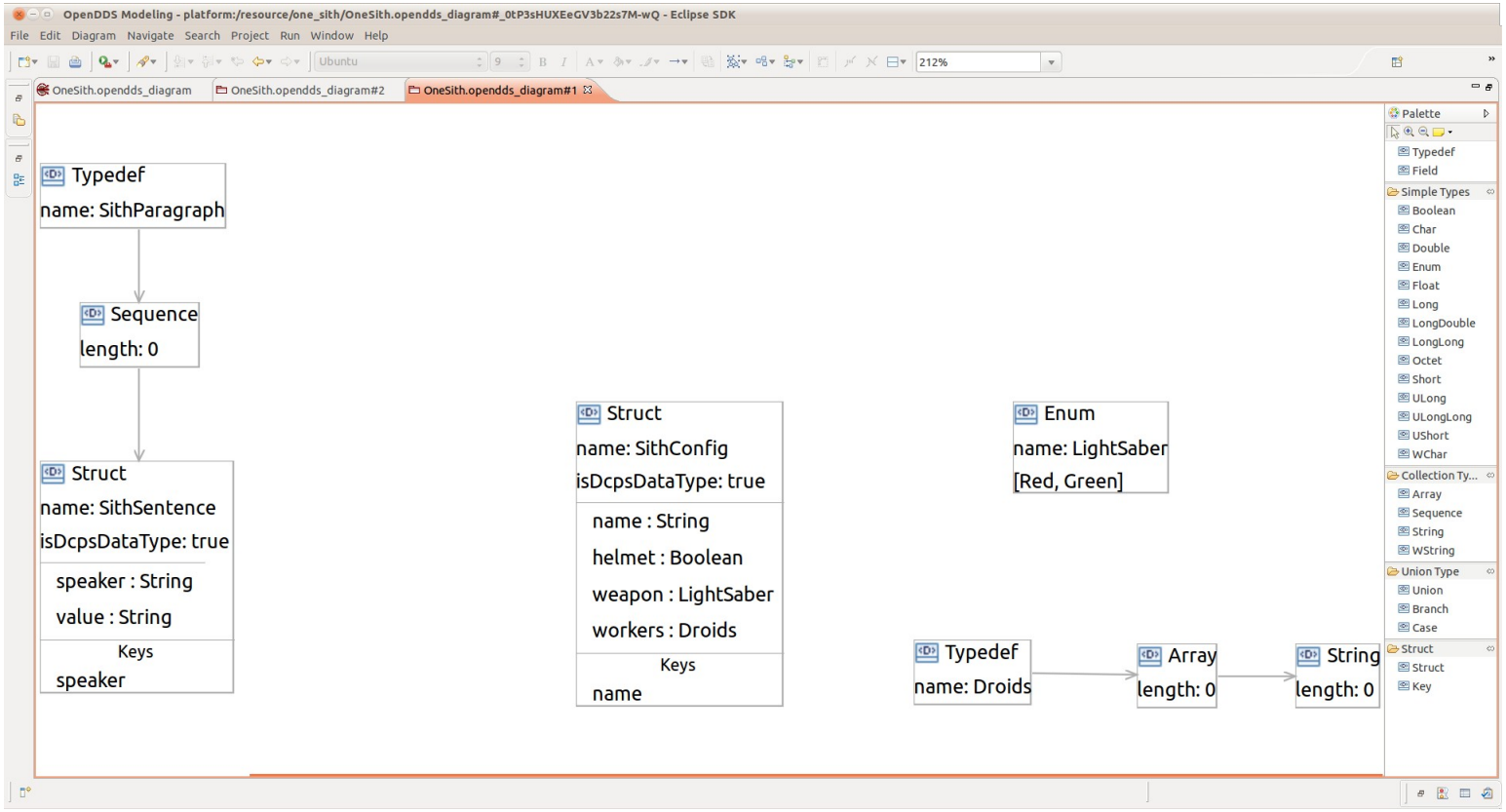
QoS Policy Editor

- Creates reusable policy values
  - Referenced by qualified name
  - Multiple values for the same policy type
- Can be referenced in any appropriate context
  - e.g. DataWriter policies in DataWriters
- Includes all specification defined policies
- Modify values using properties editor

Semantic Content



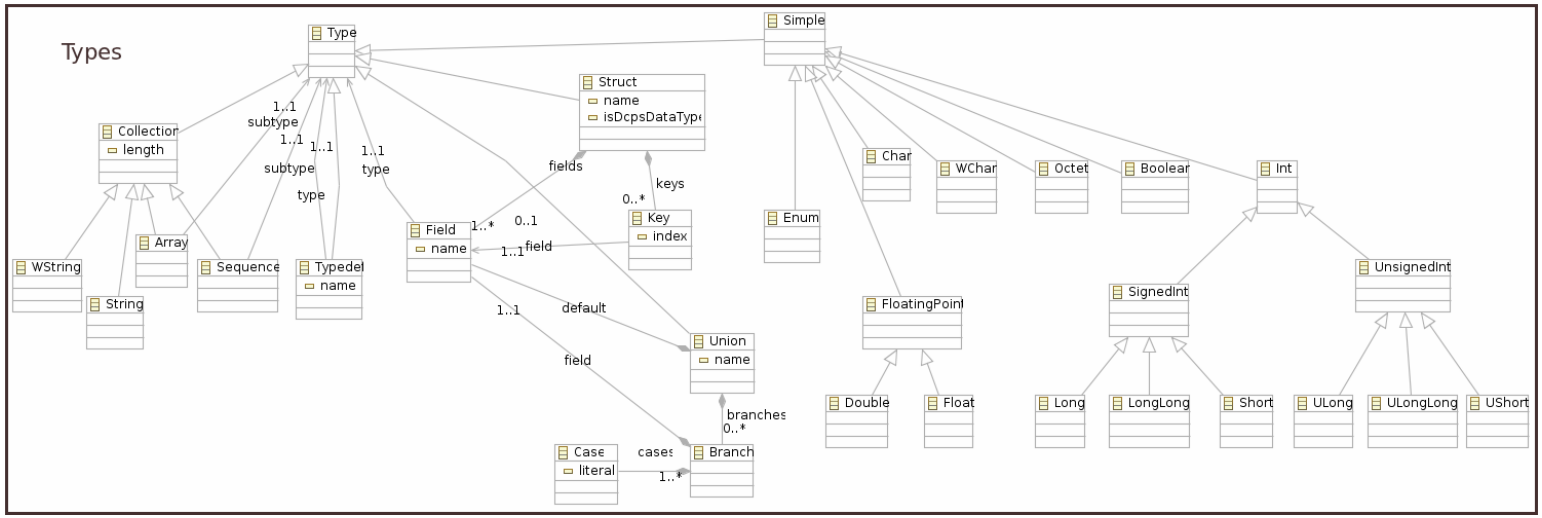
Data Definition Diagram



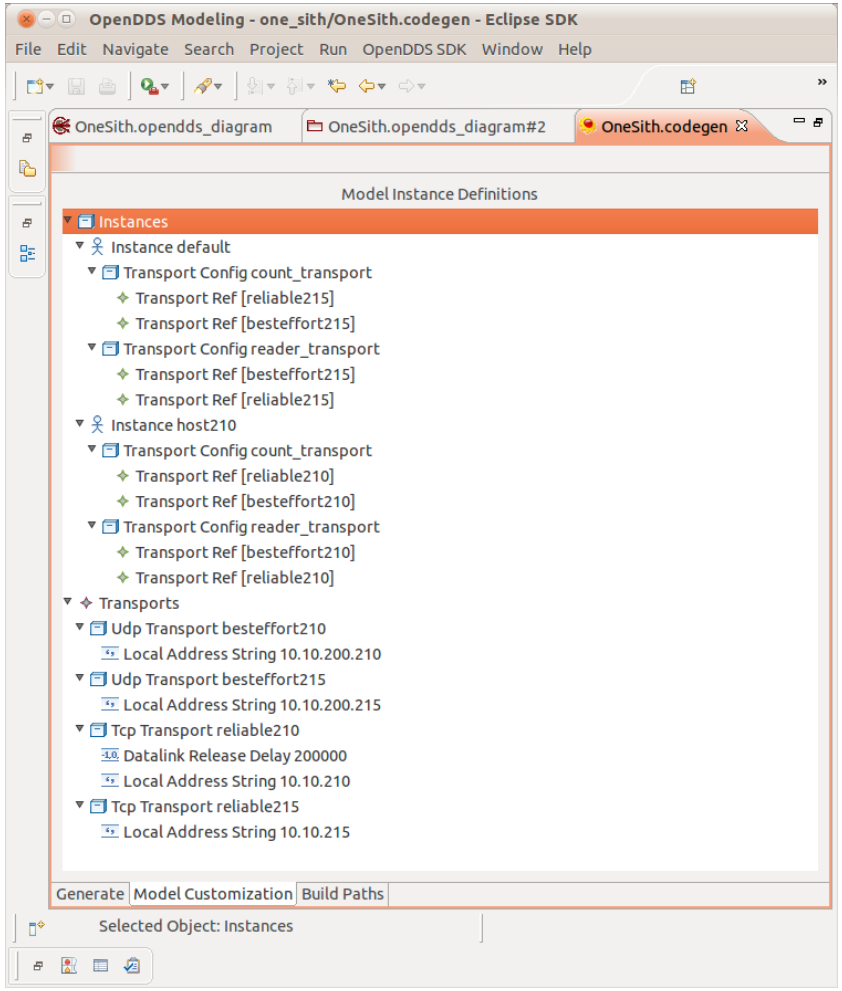
Data Definition Editor

- DDS Subset of IDL data types
  - No interfaces or exceptions
  - Simple IDL data types
    - boolean, char, double, enum, float, long, long double, long long, octet, short, unsigned long, unsigned long long, unsigned short, wchar
  - Collection data types
    - array, sequence, string, wstring
  - Complex data types
    - structures, unions
  - Also
    - typedefs
    - Structure fields can be marked as DCPS keys
- Structures can be marked for transport
  - Indicates a structure can be bound directly to a Topic
  - Causes TypeSupport code to be generated

Semantic Content



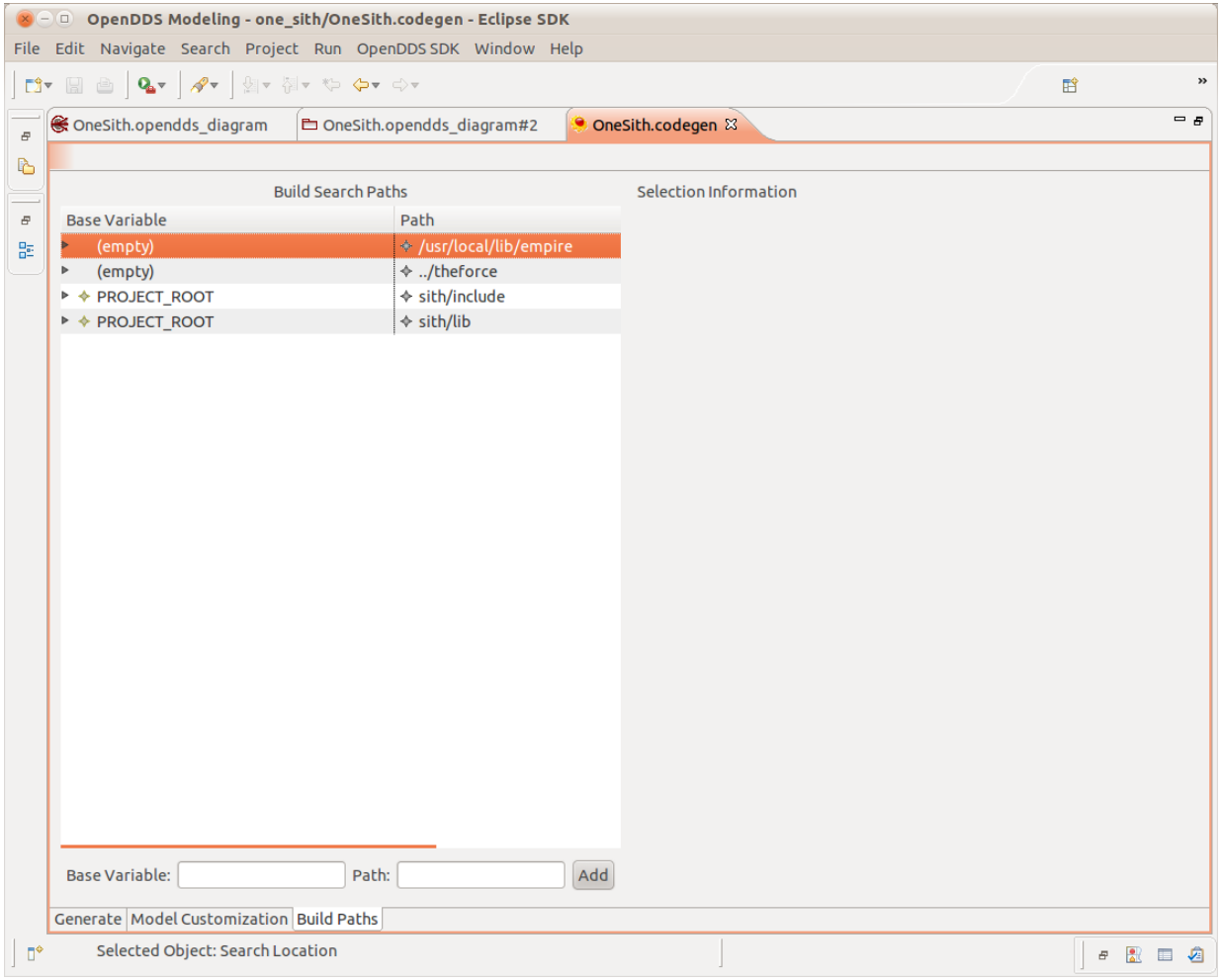
Model Customization Form



Model Customization Editor

- Allows transport details to be defined
  - Define transport characteristics to be included in lists
    - Configuration: Host addresses and ports
    - Operating behaviors: timeouts, buffer sizes, queues, threads, etc.
    - Transport specific specializations: connection retries, max packet size
- Allows 'instances' to be defined
  - Allows model to occur multiple times in same application
  - Can define deployment information
  - Includes priority ordered lists of transports
    - Bound to model using 'Config' element name as attribute value for model element

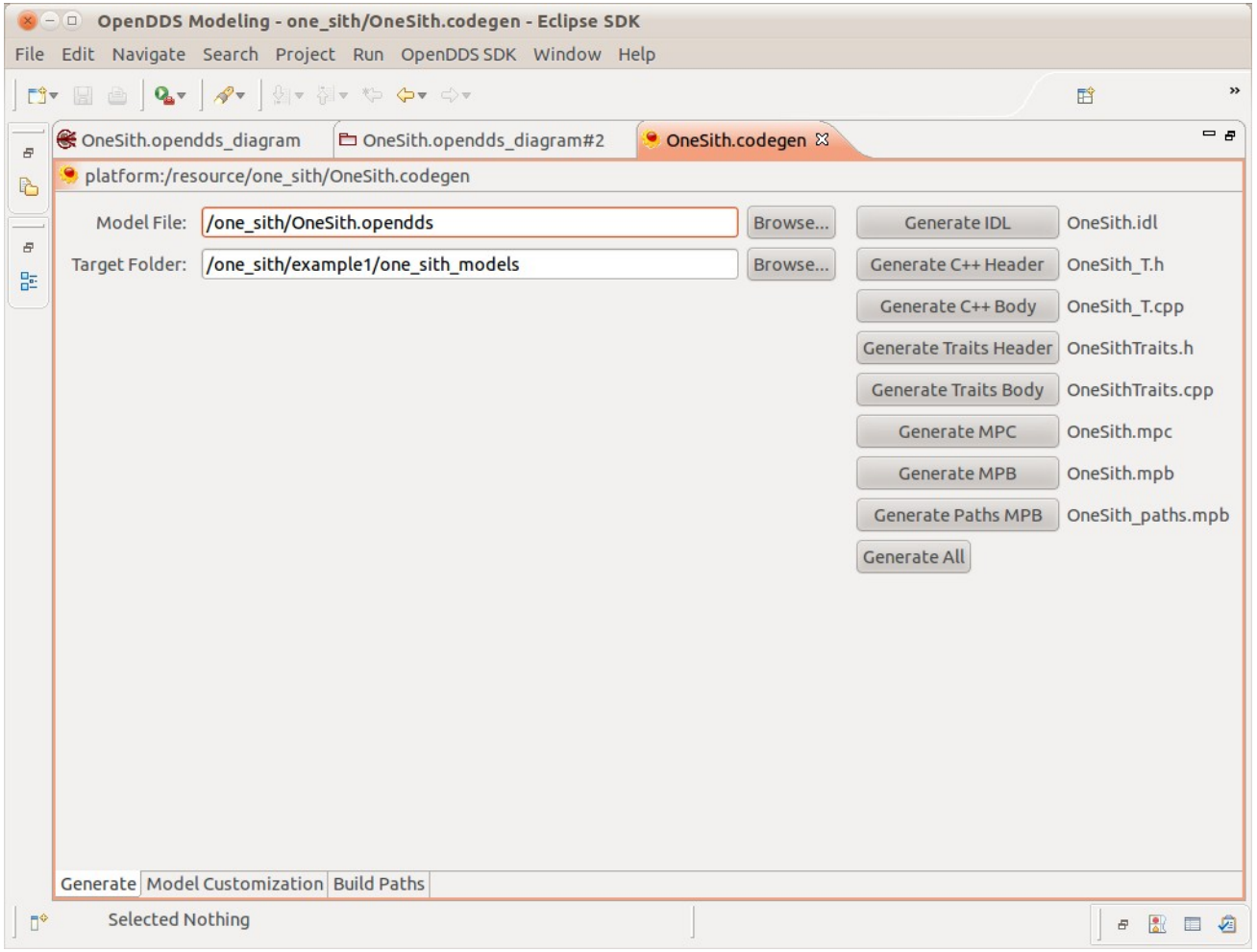
Build Support Form



Build Support Editor

- Allows search paths to be specified
  - For header files (C++ and IDL)
  - For link libraries
- Absolute paths
  - /path/to/include/dir
- Relative paths
  - path/to/include/dir
- Paths relative to build time environment
  - $PROJECT_HOME/include/dir

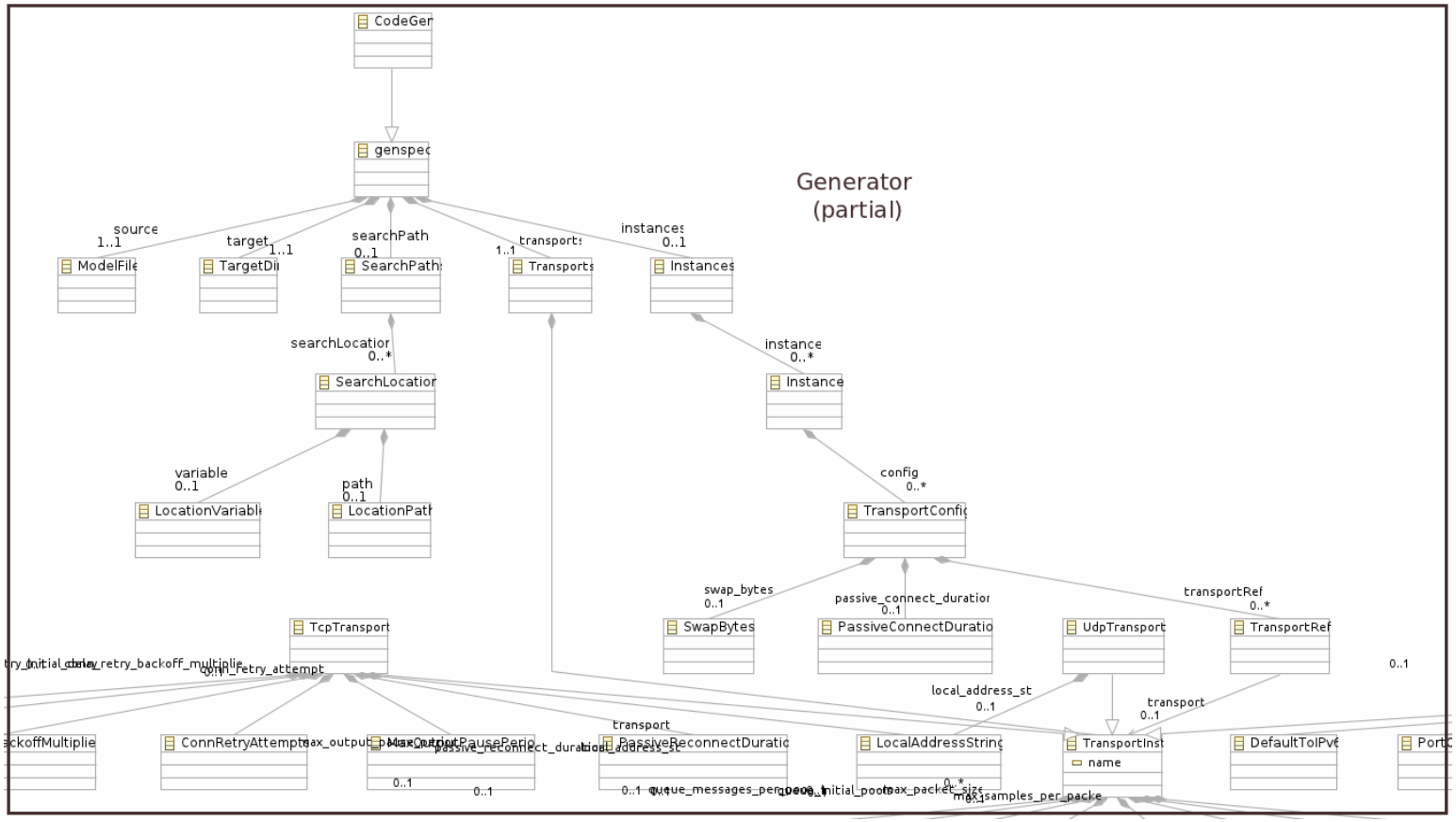
Code Generation Form



Code Generation

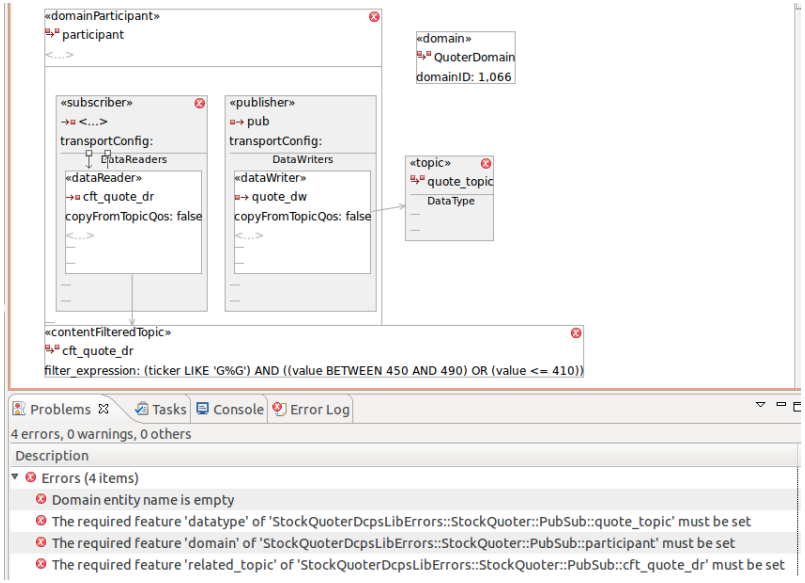
- Paths can be relative or absolute
  - Root is eclipse workspace root
  - Select source model
    - Semantic model file
  - Select target directory
    - Directory reachable in eclipse workspace
    - Can be externally linked folder (in external filesystem)
- Generated filenames shown
- Generate one or all files
  - IDL source
  - C++ header/implementation
  - Build support (mpc,mpb)
- C++ files based on support library
  - More on this on the application integration slides

Semantic Content



Validations

- XML instance document validates to XSD
- Model validates semantic rules
  - Captured using OCL
  - Follow specification constraints
  - Displayed in the “Problems” view



Annotations

- Included in semantic model
  - Attached to an element using its Properties page
  - Added as comments to generated code
    - Template and traits file comments not very useful
    - IDL comments very useful: document information within the IDL files about the defined types (graphical model not needed for notations)
    - Comments are included in code near the element the comments were attached to
    - Single line and multi-line comments allowed
  - Type of comments can be selected in the model
    - Simple comments
    - Doxygen comments (included in generated documentation using the *doxygen* application)
- Not included in semantic model
  - Placed on graphical model
    - Document information about the model on the diagrams
  - Do not appear in generated code

Model Serialization

- Model serialized to three files
  - **.opendds_diagram*
    - XMI: contains tool specific graphical information
    - Contains 'HREF' links to the XML files with the semantic content
  - **.opendds*
    - XML: contains DDS and deployment semantic information
    - Schema is available for use in creating translators and instance document validation
      - Included in the installed org.opendds.modeling.model plug-in
      - Schema located at *model/OpenDDSXMI.xsd*
      - imports other OpenDDS schemas within the same package
    - Eclipse can create an *opendds_diagram file using only this semantic content – allows importing from translated documents
  - **.codegen*
    - XML: contains deployment semantic information
    - Schema is available for use in creating translators and instance document validation
      - Included in the installed org.opendds.modeling.sdk.model plug-in
      - Schema located at *model/GeneratorXMI.xsd*

Serialized Model Fragments

- Fragment of an *.opendds_diagram file

```
<?xml version="1.0" encoding="UTF-8"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:dcps="http://www.opendds.org/modeling/schemas/DCPS/1.0" xmlns:notation="http://www.eclipse.org/gmf/runtime/1.0.2/notation"
xmlns:opendds="http://www.opendds.org/modeling/schemas/OpenDDS/1.0" xmlns:topics="http://www.opendds.org/modeling/schemas/Topics/1.0"
xmlns:types="http://www.opendds.org/modeling/schemas/Types/1.0">
  <notation:Diagram xmi:id="_n9Ew0HUXEeGV3b22s7M-wQ" type="OpenDDS" name="OneSith.opendds_diagram" measurementUnit="Pixel">
    <children xmi:type="notation:Shape" xmi:id="_pq1TIHUXEeGV3b22s7M-wQ" type="2010" fontName="Ubuntu">
      <children xmi:type="notation:DecorationNode" xmi:id="_pq4WcHUXEeGV3b22s7M-wQ" type="5019"/>
      <children xmi:type="notation:BasicCompartment" xmi:id="_pq49gHUXEeGV3b22s7M-wQ" type="7001">
        <children xmi:type="notation:Shape" xmi:id="_yUaYEHUXEeGV3b22s7M-wQ" type="3003" fontName="Ubuntu">
          <children xmi:type="notation:DecorationNode" xmi:id="_yUa_IHUXEeGV3b22s7M-wQ" type="5022"/>
          <children xmi:type="notation:DecorationNode" xmi:id="_yUa_IXUXEeGV3b22s7M-wQ" type="5023"/>
          <styles xmi:type="notation:HintedDiagramLinkStyle" xmi:id="_yUaYEXUXEeGV3b22s7M-wQ" diagramLink="_JK5noHUYEeGV3b22s7M-wQ" hint="OpenDDS DcpsLib"/>
          <element xmi:type="opendds:DcpsLib" href="OneSith.opendds#_yUZJ8HUXEeGV3b22s7M-wQ"/>
          <layoutConstraint xmi:type="notation:Bounds" xmi:id="_yUaYEnUXEeGV3b22s7M-wQ" x="32" y="30"/>
        </children>
        <children xmi:type="notation:Shape" xmi:id="_AFI6wHg-EeGPHbuzAuI2wQ" type="3004" fontName="Ubuntu">
          <children xmi:type="notation:DecorationNode" xmi:id="_AFJh0Hg-EeGPHbuzAuI2wQ" type="5024"/>
          <children xmi:type="notation:DecorationNode" xmi:id="_AFJh0Xg-EeGPHbuzAuI2wQ" type="5025"/>
          <styles xmi:type="notation:HintedDiagramLinkStyle" xmi:id="_AFI6wXg-EeGPHbuzAuI2wQ" diagramLink="_BYLP8Hg-EeGPHbuzAuI2wQ" hint="OpenDDS PolicyLib"/>
          <element xmi:type="opendds:PolicyLib" href="OneSith.opendds#_AEnWUXg-EeGPHbuzAuI2wQ"/>
          <layoutConstraint xmi:type="notation:Bounds" xmi:id="_AFI6wng-EeGPHbuzAuI2wQ" x="188" y="30"/>
        </children>
```

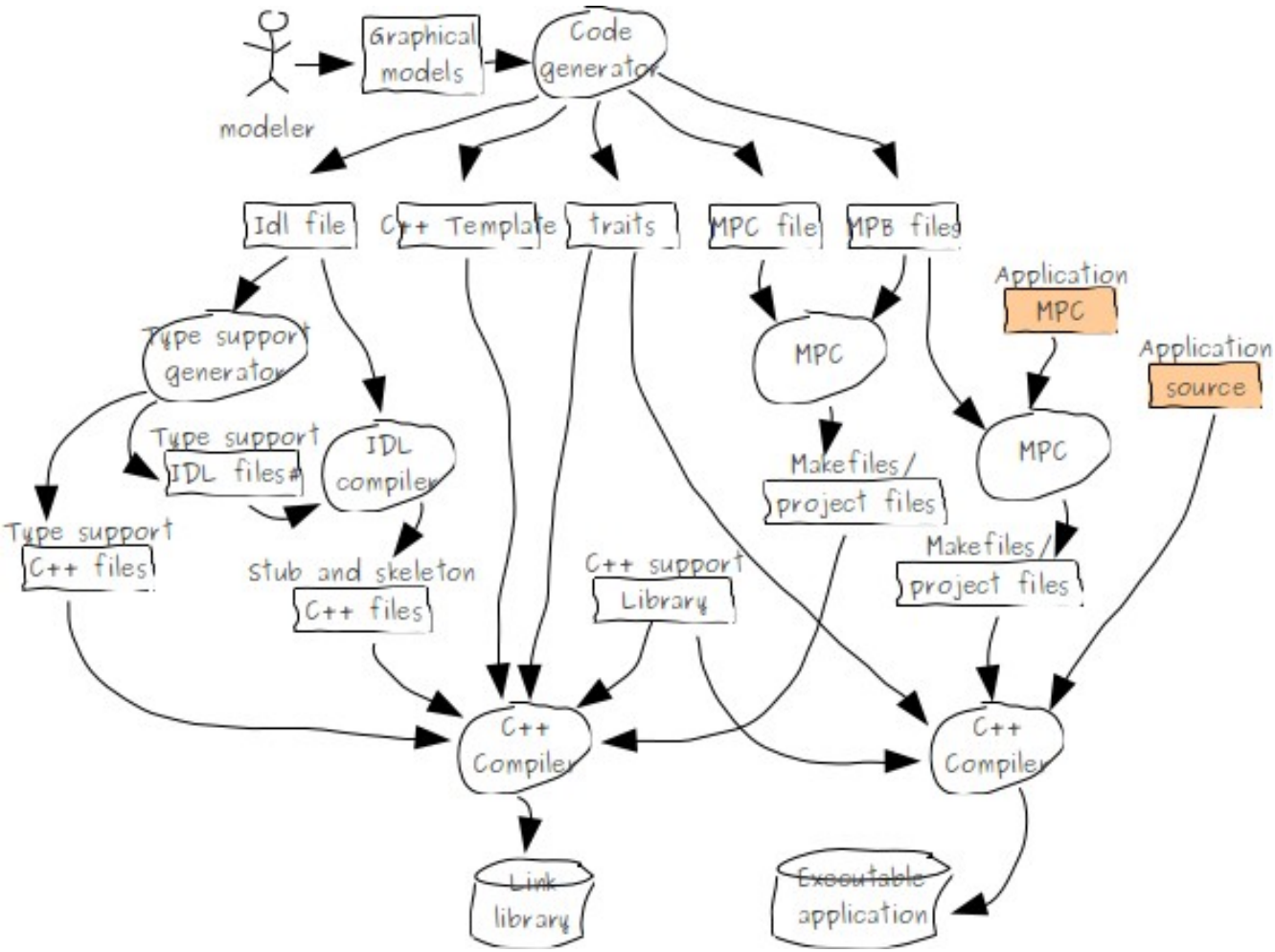
- Fragment of an *.opendds file

```
<?xml version="1.0" encoding="UTF-8"?>
<opendds:OpenDDSModel xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:opendds="http://www.opendds.org/modeling/schemas/OpenDDS/1.0" xmlns:topics="http://www.opendds.org/modeling/schemas/Topics/1.0"
xmlns:types="http://www.opendds.org/modeling/schemas/Types/1.0" xmi:id="_n8OcQXUXEeGV3b22s7M-wQ" name="OneSith">
  <packages xmi:id="_pqacYHUXEeGV3b22s7M-wQ" name="Empire">
    <libs xsi:type="opendds:DcpsLib" xmi:id="_yUZJ8HUXEeGV3b22s7M-wQ" name="Masters">
      <domains xmi:id="_M2tyEHUYEeGV3b22s7M-wQ" name="empire" domainId="666"/>
      <participants xmi:id="_Jq6kcHUYEeGV3b22s7M-wQ" name="dooku" transportConfig="count_transport" domain="_M2tyEHUYEeGV3b22s7M-wQ">
        <publishers xsi:type="opendds:publisher" xmi:id="_UQyW8HUYEeGV3b22s7M-wQ" name="dooku_pub" transportId="-1">
          <writers xmi:id="_UQy-AHUYEeGV3b22s7M-wQ" name="dooku_writer" durability="_RE_HEHg-EeGPHbuzAuI2wQ" history="_ioO-0HUyEeGV3b22s7M-wQ"
copyFromTopicQos="false" topic="_WKFa4HUYEeGV3b22s7M-wQ"/>
        </publishers>
        <subscribers xsi:type="opendds:subscriber" xmi:id="_VW5qcHUYEeGV3b22s7M-wQ" name="dooku_sub" transportId="-1">
          <readers xmi:id="_VW5qcXUYEeGV3b22s7M-wQ" name="dooku_reader" transportConfig="reader_transport" durability="_t1oEUHUyEeGV3b22s7M-wQ"
history="_t16_QHUyEeGV3b22s7M-wQ" copyFromTopicQos="false" topic="_WKFa4HUYEeGV3b22s7M-wQ"/>
        </subscribers>
      </participants>
      <topicDescriptions xsi:type="topics:Topic" xmi:id="_WKFa4HUYEeGV3b22s7M-wQ" name="SpokenSith" datatype="_1WqBEHUXEeGV3b22s7M-wQ"/>
      <policies xsi:type="opendds:historyQosPolicy" xmi:id="_ioO-0HUyEeGV3b22s7M-wQ" name="history" depth="500" kind="KEEP_LAST"/>
      <policies xsi:type="opendds:durabilityQosPolicy" xmi:id="_t1oEUHUyEeGV3b22s7M-wQ" name="durability" kind="TRANSIENT_LOCAL"/>
      <policies xsi:type="opendds:historyQosPolicy" xmi:id="_t16_QHUyEeGV3b22s7M-wQ" name="history" depth="1" kind="KEEP_ALL"/>
    </libs>
```

Application Integration

- Code generated from model
  - IDL PSM: IDL file is generated for data definitions
  - DDS Entities: C++ template specializing support library with model specific information
  - Traits: application interface to model elements
  - Build files:
    - MPC file to generate link library from model
    - MPB files to link model library into application
- Libraries built from generated code
  - One library from each model
  - Link with application and support libraries
- Support libraries
  - Provide standardized interface to the DDS API Entities
  - Encapsulate OpenDDS service lifetime and Entity management

Build Process



Support Libraries

- Built as part of OpenDDS
  - Separate, conditionally compilable link library and headers
- Encapsulates OpenDDS specific features
  - Service startup and lifetime management
- Encapsulates Middleware lifetime management
  - Accessed through generated *traits.h header file
    - Defines model types that can be instantiated
  - Delegated to from generated template *_T.{h,cpp} files
    - Provides efficient access to all model defined DDS Entities
    - Lazy initialization of Entities and required precursors (containers)
- Provides utilities
  - Null implementations for listeners
  - Common synchronization patterns
    - Useful when starting or terminating to coordinate decoupled applications

Application Code

- OpenDDS::Model::Application
  - Object encapsulates OpenDDS service
  - Lifetime of the object is the lifetime of the service
- Empire::Masters::DefaultOneSithType
  - Object encapsulates middleware model
  - Lifetime of model is the lifetime of the middleware Entities
  - Defined in *OneSithTraits.h* header file
  - Accessors for Entities using enumerations

```
#include "OneSithTraits.h"
...
int
main( int argc, char** argv, char**)
{
  try {
    OpenDDS::Model::Application app( argc, argv);
    Empire::Masters::DefaultOneSithType model( app, argc, argv);
...
    using OpenDDS::Model::Empire::Masters::Elements;
    DDS::DataReader_var reader = model.reader( Elements::DataReaders::dooku_reader);
...
    DDS::DataWriter_var writer = model.writer( Elements::DataWriters::dooku_writer);
    EmpireData::SithSentenceDataWriter_var sith_writer
      = EmpireData::SithSentenceDataWriter::_narrow(writer);
...
  } catch( std::exception& ex) {
    std::cerr << "Exception caught in main(): " << ex.what() << std::endl;
    return -1;
  }
  return 0;
}
```

Single Application Model

- Single model
  - Includes all middleware
  - Includes all data definitions
  - Includes all QoS definitions
- Single customization
  - Single instance
- Single target
  - One link library
  - Contains all model code
- Single Application
  - Links with model library
  - Integrates with external system/applications

Multiple Application Model

- Single model
  - Includes all middleware
  - Includes all data definitions
  - Includes all QoS definitions
- One or more customizations
  - One or more instances
- Single target
  - One link library
  - Contains all model code
- Multiple Applications
  - Each links with same model library
  - Only access Entities needed in specific application

Shared Models

- Multiple models
  - Grouped logically into libraries
  - Can share data definitions
  - Can share QoS definitions
  - Can share Topic definitions
- One or more customizations
  - One or more instances
  - As the specific model requires
- Multiple targets
  - Multiple link libraries
- Multiple Applications
  - Each links with one or more model libraries
  - Only access Entities needed in specific application

Next Steps

- Migrate to new UML Profile for DDS when adopted
- Migrate to new C++ mapping when adopted
- Incorporate recent RTPS capabilities in customization forms
- Package as RCP for version management
- Import from existing models
- DLRL?
- Secure DDS?
- FACE (Navy)?