Modelica Association Project
“Distributed Co-Simulation Protocol”

Martin Krammer
DCP MAP Leader
martin.krammer@v2c2.at
Outline

- Introduction
- The Distributed Co-Simulation Protocol (DCP)
  - Communication Protocol
  - Architecture Description
  - Operating Modes
  - State Machine
  - Exchange of Input and Output Data
  - Use Case
- The Future of DCP
Motivation

- The Functional Mock-up Interface (FMI, MODELISAR project) standardizes integration of simulation models, tools and solvers.
- But what about distributed setups?

- Until now, this is done manually.
The Functional Mock-up Interface (FMI, MODELISAR project) standardizes integration of simulation models, tools and solvers.

But what about distributed setups?

Until now, this is done manually.
Motivation

- The Functional Mock-up Interface (FMI, MODELISAR project) standardizes integration of simulation models, tools and solvers.
- But what about distributed setups?

- Until now, this is done manually.
Motivation

- The Functional Mock-up Interface (FMI, MODELISAR project) standardizes integration of simulation models, tools and solvers.
- But what about distributed setups?

- Until now, this is done manually.
The ACOSAR project

– *Advanced Co-Simulation Open System Architecture*
– Duration: 09/2015-08/2018
– Costs: 8,123k€
– Effort: 60 PY

ACOSAR focuses on integration of

– Real-time and real-time, and
– Real-time and non-real-time systems

**Primary goal:** Negotiate technical specification of communication protocol intended for standardization.
The Distributed Co-Simulation Protocol

Main design aspects

- Interoperability
  - Define a communication protocol
  - Goal: Pursue standardization with a recognized standardization body

- Compatibility
  - Support a broad range of systems, from small microcontrollers to large test rigs
  - Targets: Low overhead, low memory footprint

- Integration
  - Develop methodology for application in development processes
  - Master-Slave concept

- Communication
  - Support multiple transport protocols
  - Initially: UDP, CAN, USB, Bluetooth, and EtherCAT

- Economy
  - Reduce development time
  - Decrease computing cost
  - Accelerate time-to-market
The Distributed Co-Simulation Protocol

- Default integration methodology
- Relies on DCP slave description file (.dcpx)
- Defines provider-integrator relationship
The Distributed Co-Simulation Protocol

- Default integration methodology
- Relies on DCP slave description file (.dcpx)
- Defines provider-integrator relationship
The Distributed Co-Simulation Protocol

- **DCP Slave Description**
  - DCP slave description accompanies DCP slave
  - Specified as XSD 1.1 schema definition
  - XML instance .dcpx File

- **Assertions**
  - Used to enforce specifications
  - Avoid incorrect definitions

- **Available transformation strips**
  - assertions and generates XSD 1.0 schema

Diagram:

- `dcuSlaveDescription`
- `attributes`
- `OpMode`
- `UnitDefinitions`
- `TypeDefinitions`
- `VendorAnnotations`
- `TimeRes`
- `Heartbeat`
- `Drivers`
- `CapabilityFlags`
- `Variables` (inputs/outputs/parameters)
- `Logging definitions`

- Operating mode
- Units
- Types
- Vendor specific annotations
- Time resolution
- Heartbeat definitions
- Transport protocol
- Capability flags
- Variables (inputs/outputs/parameters)
- Logging definitions
The Distributed Co-Simulation Protocol

Architecture Description

- DCP Master/Slave
  - HiL system, test bench, simulation tool, ...
  - API (vendor-specific)
  - Standardized DCP data model (XML schema files)
  - Middleware (vendor-specific)
  - DCP protocol
    - Bluetooth/RFCOMM
    - UDP/IPv4
    - USB
    - CAN

Communication protocol
Transport protocol

native DCP
non-native DCP
The Distributed Co-Simulation Protocol

Architecture Description

- DCP Master/Slave
  - HiL system, test bench, simulation tool, ...
  - API (vendor-specific)
  - Standardized DCP data model (XML schema files)
  - Middleware (vendor-specific)

Communication protocol
- Transport protocol

DCP protocol
- Bluetooth/RFCOMM
- UDP/IPv4
- USB
- CAN

native DCP
non-native DCP
The Distributed Co-Simulation Protocol

Architecture Description

Diagram:

- Communication protocol
- DCP PDU
  - Standardized mapping
  - Non-standardized mapping
- Transport protocol
  - Payload: DCP PDU
  - Transport protocol PDU incl. DCP PDU
- native DCP
- non-native DCP

Martin Krammer, VIF
The Distributed Co-Simulation Protocol

- **Taxonomy of Protocol Data Units (PDU)**
  - "PDU Families"

<table>
<thead>
<tr>
<th>Configuration (CFG)</th>
<th>State change (STC)</th>
<th>Notification (INF)</th>
<th>Response (RSP)</th>
<th>Data (DAT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFG_set_time_res</td>
<td>STC_register</td>
<td>INF_state</td>
<td>RSP_ack</td>
<td>DAT_input_output</td>
</tr>
<tr>
<td>CFG_set_steps</td>
<td>STC_unregister</td>
<td>INF_error</td>
<td>RSP_nack</td>
<td>DAT_parameter</td>
</tr>
<tr>
<td>CFG_config_output</td>
<td>STC_config</td>
<td>INF_log</td>
<td>RSP_stateAck</td>
<td></td>
</tr>
<tr>
<td>CFG_config_clear</td>
<td>STC_initialize</td>
<td></td>
<td>RSP_errorAck</td>
<td></td>
</tr>
<tr>
<td>CFG_set_target_net_info</td>
<td>STC_run</td>
<td></td>
<td>RSP_logAck</td>
<td></td>
</tr>
<tr>
<td>CFG_set_source_net_info</td>
<td>STC_reset</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CFG_set_parameter</td>
<td>STC_send_outputs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CFG_set_configurable_parameter</td>
<td>STC_stop</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CFG_set_network_info</td>
<td>STC_do_step</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Protocol Data Unit (PDU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-Way</td>
</tr>
<tr>
<td>Data PDU</td>
</tr>
<tr>
<td>Request PDU</td>
</tr>
<tr>
<td>Configuration</td>
</tr>
<tr>
<td>Information</td>
</tr>
<tr>
<td>Two-Way</td>
</tr>
<tr>
<td>Data PDU</td>
</tr>
<tr>
<td>Response PDU</td>
</tr>
<tr>
<td>Notification PDU</td>
</tr>
<tr>
<td>State Machine</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Control Protocol (DCP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCP Master</td>
</tr>
<tr>
<td>DCP Slave</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs/Outputs</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data (DAT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notification (NTF)</td>
</tr>
<tr>
<td>Request (RSP)</td>
</tr>
<tr>
<td>Configuration (CFG)</td>
</tr>
<tr>
<td>State change (STC)</td>
</tr>
<tr>
<td>Information (INF)</td>
</tr>
</tbody>
</table>

**DCP Frame**
- Data (DAT)
- Protocol Data Unit (PDU)
- Control Protocol (DCP)

**Type Identifiers**
- **PDU Families**
- **PDU Identities**

**Protocol Identities**
- **Protocol Identities**
- **Protocol Identities**

**Control Protocols**
- **Control Protocols**
- **Control Protocols**

**Notation**
- **Notation**
- **Notation**

**Diagram**
- **Diagram**
- **Diagram**

**Table**
- **Table**
- **Table**

**Legend**
- **Legend**
- **Legend**

**References**
- **References**
- **References**
The Distributed Co-Simulation Protocol

**Operating Modes**
- The DCP covers three different time domains

<table>
<thead>
<tr>
<th>Operating mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soft real-time (SRT)</td>
<td>Synchronous to absolute time, tolerant to RT violations</td>
</tr>
<tr>
<td>Hard real-time (HRT)</td>
<td>Synchronous to absolute time, intolerant to RT violations</td>
</tr>
<tr>
<td>Non-real-time (NRT)</td>
<td>Independent of absolute time</td>
</tr>
</tbody>
</table>
The Distributed Co-Simulation Protocol

- DCP slave state machine for simulation control
- A typical simulation cycle
  1. Registration
  2. Configuration
  3. Initialization
  4. Run/Compute
  5. Stop
  6. (Error)

Image source: DCP specification v1.0 RC 2
The Distributed Co-Simulation Protocol

- DCP slave state machine for simulation control
- A typical simulation cycle
  1. Registration
  2. Configuration
  3. Initialization
  4. Run/Compute
  5. Stop
  6. (Error)

Image source: DCP specification v1.0 RC 2
The Distributed Co-Simulation Protocol

- DCP slave state machine for simulation control
- A typical simulation cycle
  1. Registration
  2. Configuration
  3. Initialization
  4. Run/Compute
  5. Stop
  6. (Error)

Image source: DCP specification v1.0 RC 2
The Distributed Co-Simulation Protocol

- DCP slave state machine for simulation control
- A typical simulation cycle
  1. Registration
  2. Configuration
  3. Initialization
  4. Run/Compute
  5. Stop
  6. (Error)

Image source: DCP specification v1.0 RC 2
The Distributed Co-Simulation Protocol

- DCP slave state machine for simulation control
- A typical simulation cycle
  1. Registration
  2. Configuration
  3. Initialization
  4. Run/Compute
  5. Stop
  6. (Error)
The Distributed Co-Simulation Protocol

- DCP slave state machine for simulation control
- A typical simulation cycle
  1. Registration
  2. Configuration
  3. Initialization
  4. Run/Compute
  5. Stop
  6. (Error)

Image source: DCP specification v1.0 RC 2
The Distributed Co-Simulation Protocol

- DCP slave state machine for simulation control
- A typical simulation cycle
  1. Registration
  2. Configuration
  3. Initialization
  4. Run/Compute
  5. Stop
  6. (Error)

Image source: DCP specification v1.0 RC 2
The Distributed Co-Simulation Protocol

- Exchange of data during simulation phase
  1. Configuration must be generated by DCP master
  2. Configuration must be rolled out to DCP slaves prior to simulation

- Zero run time overhead during simulation

- Example:
The Distributed Co-Simulation Protocol

- Exchange of data during simulation phase
  1. Configuration must be generated by DCP master
  2. Configuration must be rolled out to DCP slaves prior to simulation
- Zero run time overhead during simulation
- Example:

```plaintext
DCP slave 1: 192.168.2.5
vr = 1
dt = uint8

DCP slave 2: 192.168.2.7
vr = 2
dt = float32
```
The Distributed Co-Simulation Protocol

- Exchange of data during simulation phase
  1. Configuration must be generated by DCP master
  2. Configuration must be rolled out to DCP slaves prior to simulation
- Zero run time overhead during simulation
- Example:

```
DCP master
192.168.2.2

DCP slave 1
192.168.2.5
vr = 1
dt = uint8

DCP slave 2
192.168.2.7
vr = 2
dt = float32
```
The Distributed Co-Simulation Protocol

- Exchange of data during simulation phase
  1. Configuration must be generated by DCP master
  2. Configuration must be rolled out to DCP slaves prior to simulation
- Zero run time overhead during simulation
- Example:

```
DCP master
192.168.2.2

DCP slave 1
192.168.2.5
vr = 1
dt = uint8

DCP slave 2
192.168.2.7
vr = 2
dt = float32
```

Martin Krammer, VIF
The Distributed Co-Simulation Protocol

- Exchange of data during simulation phase
  1. Configuration must be generated by DCP master
  2. Configuration must be rolled out to DCP slaves prior to simulation

- Zero run time overhead during simulation

Example:

1. Configuration must be generated by DCP master.
2. Configuration must be rolled out to DCP slaves prior to simulation.

**Example:**

```
1. Configuration output to DCP slave 1:
   type_id = 0x23,
   pdu_seq_id = 1,
   receiver = 1,
   data_id = 0,
   pos = 0,
   source_vr = 1

2. Configuration output to DCP slave 2:
   type_id = 0x23,
   pdu_seq_id = 2,
   receiver = 1,
   data_id = 0,
   pos = 1,
   source_vr = 2
```
The Distributed Co-Simulation Protocol

- Exchange of data during simulation phase
  1. Configuration must be generated by DCP master
  2. Configuration must be rolled out to DCP slaves prior to simulation

- **Zero run time overhead** during simulation

- Example:

```
DCP master 192.168.2.2

<table>
<thead>
<tr>
<th>DCP slave 1</th>
<th>vr = 1</th>
<th>dt = uint8</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.2.5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DCP slave 2</th>
<th>vr = 1</th>
<th>dt = uint8</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.2.7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DCP slave 2</th>
<th>vr = 2</th>
<th>dt = float32</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.2.7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3. CFG_set_target_net work_information
   type_id = 0x25,
   pdu_seq_id = 3,
   receiver = 1,
   data_id = 0,
   transport_protocol = UDP,
   target_ip_address = 192.168.2.5,
   target_port = 2048
```
The Distributed Co-Simulation Protocol

- Exchange of data during simulation phase
  1. Configuration must be generated by DCP master
  2. Configuration must be rolled out to DCP slaves prior to simulation

- Zero run time overhead during simulation

Example:
The Distributed Co-Simulation Protocol

- Exchange of data during simulation phase
  1. Configuration must be generated by DCP master
  2. Configuration must be rolled out to DCP slaves prior to simulation

- Zero run time overhead during simulation

- Example:

```
DCP master 192.168.2.2

DCP slave 1 192.168.2.5
vr = 1
dt = uint8

DCP slave 2 192.168.2.7
vr = 2
dt = float32

CFG_set_source_network_information

type_id = 0x26,
pdu_seq_id = 3,
receiver = 2,
data_id = 0,
transport_protocol = UDP,
source_port = 2049
```
The Distributed Co-Simulation Protocol

- Exchange of data during simulation phase
  1. Configuration must be generated by DCP master
  2. Configuration must be rolled out to DCP slaves prior to simulation

- Zero run time overhead during simulation

- Example:

  ![Diagram showing communication between DCP master and slaves with example data types and IDs.]

  Configuration for DCP slaves:
  - DCP slave 1 (192.168.2.5):
    - vr = 1
    - dt = uint8
  - DCP slave 2 (192.168.2.7):
    - vr = 2
    - dt = float32

  Configuration for DCP master:
  - vr = 1
  - dt = uint8

  Example payload:
  - type_id = 0xF0
  - pdu_seq_id = 1
  - data_id = 0
  - payload:
    - uint8
    - float32
The Distributed Co-Simulation Protocol

- Use case by dSPACE, RWTH Aachen, ESI-ITI

Setup on display at final event
Future of DCP

- The DCP 1.0-RC1 was submitted to Modelica Association for standardization
- Will be maintained as Modelica Association Project (MAP)
- Sustainable ACOSAR project result
  - The DCP will be freely available
  - Open for everyone!
- Website: [www.dcp-standard.org](http://www.dcp-standard.org)
“The Distributed Co-Simulation Protocol for the Integration of Real-Time Systems and Simulation Environments”


- “Requirements Engineering for Consensus-Oriented Written Technical Specifications”
  Martin Krammer, Nadja Marko and Martin Benedikt, accepted for publication at 26th IEEE International Requirements Engineering Conference, August 20-24, Banff, Alberta, Canada

- “Master for Simulation Control using the Distributed Co-Simulation Protocol”
  Martin Krammer, Martin Benedikt, accepted for publication at IEEE 16th International Conference on Industrial Informatics, July 18-20, Porto, Portugal

- “Configuration of Slaves Based on the Distributed Co-Simulation Protocol”
  Martin Krammer, Martin Benedikt, accepted for publication at 23rd International Conference on Emerging Technologies and Factory Automation, September 4th - 7th, 2018, Torino, Italy
Any questions?

Martin Krammer
martin.krammer@v2c2.at