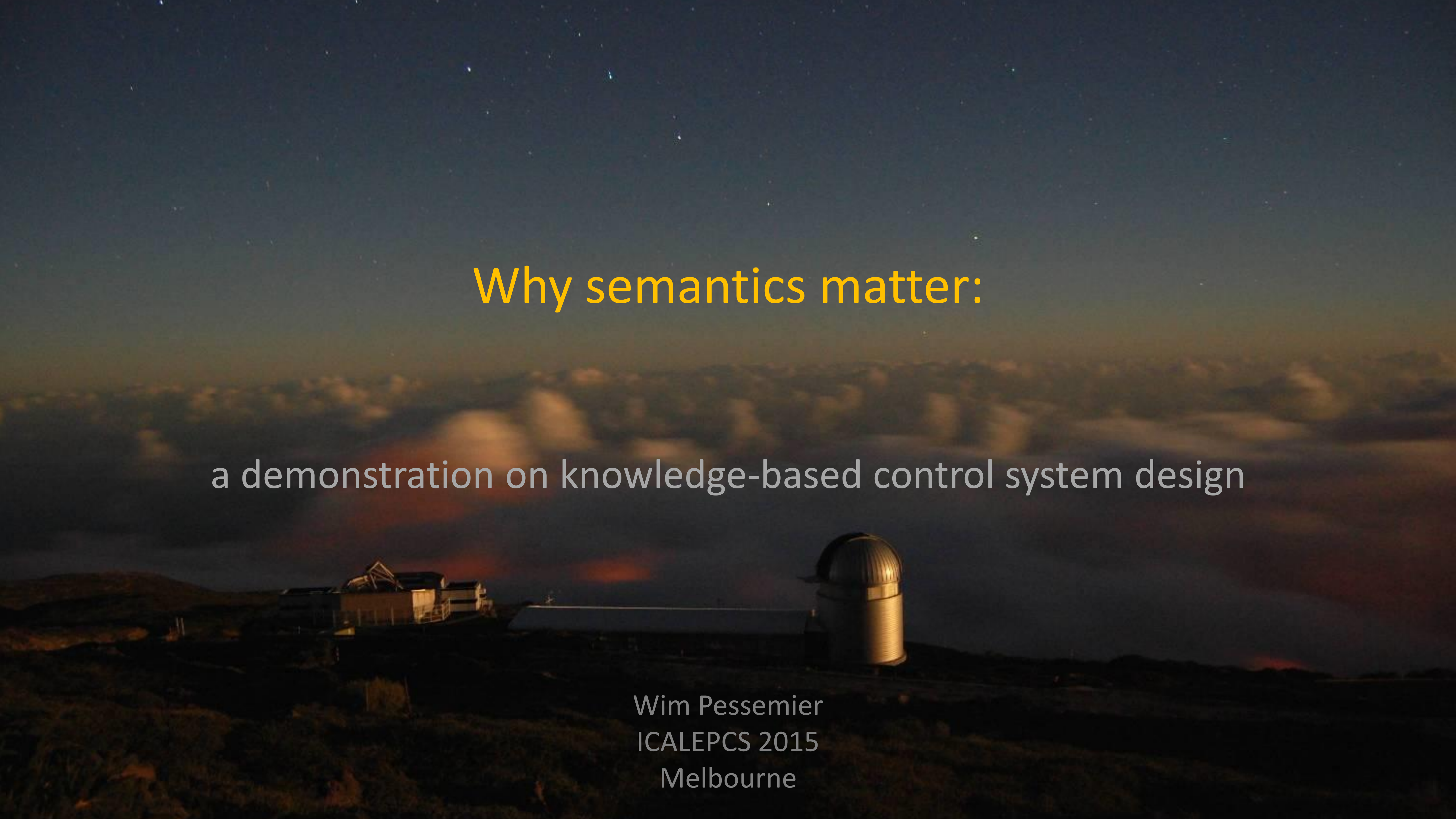
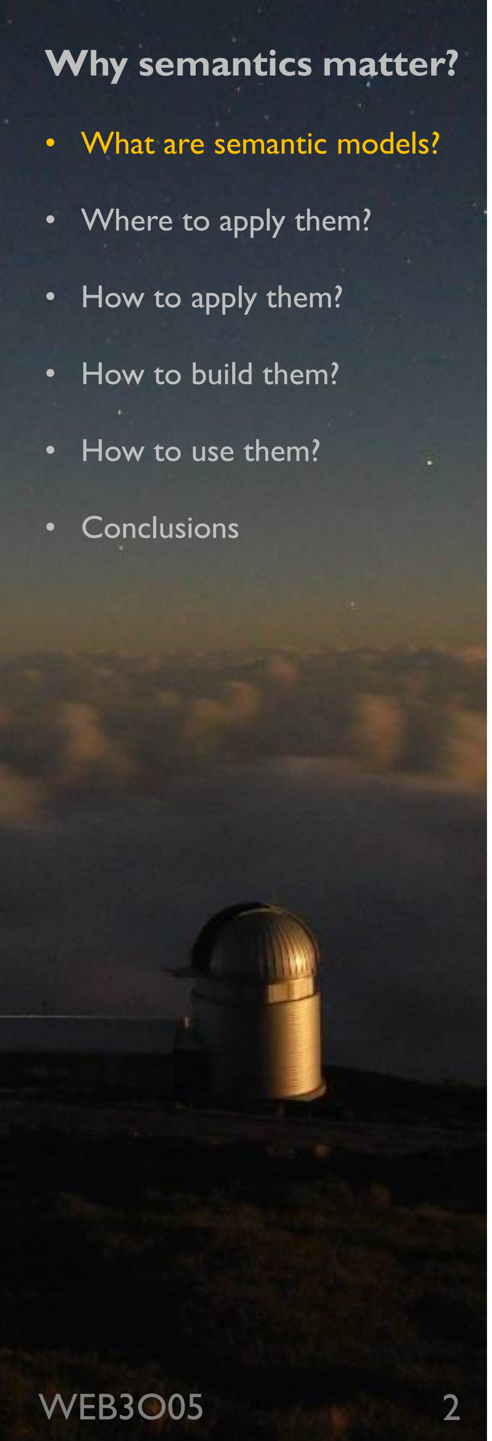


Why semantics matter:

a demonstration on knowledge-based control system design

Wim Pessemier
ICALEPCS 2015
Melbourne



Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

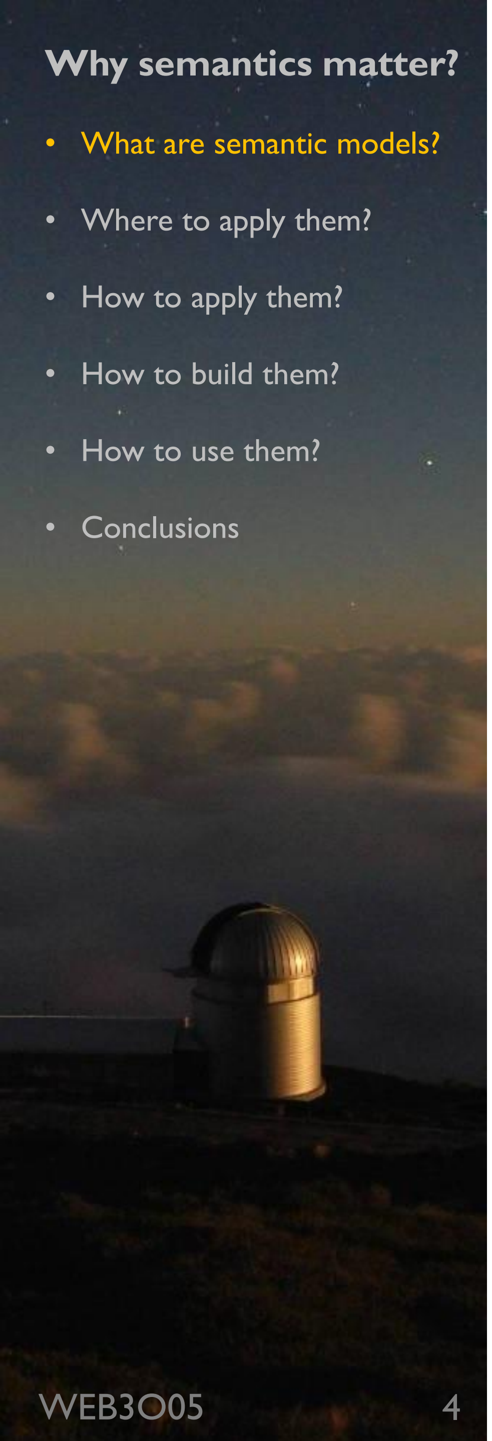
What are semantic models?

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

What are semantic models?

- Models that describe
  - pieces of information (data, descriptions)
  - their relations



Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

What are semantic models?

- Models that describe
  - pieces of information (data, descriptions)
  - their relations → **meaning (semantics)**

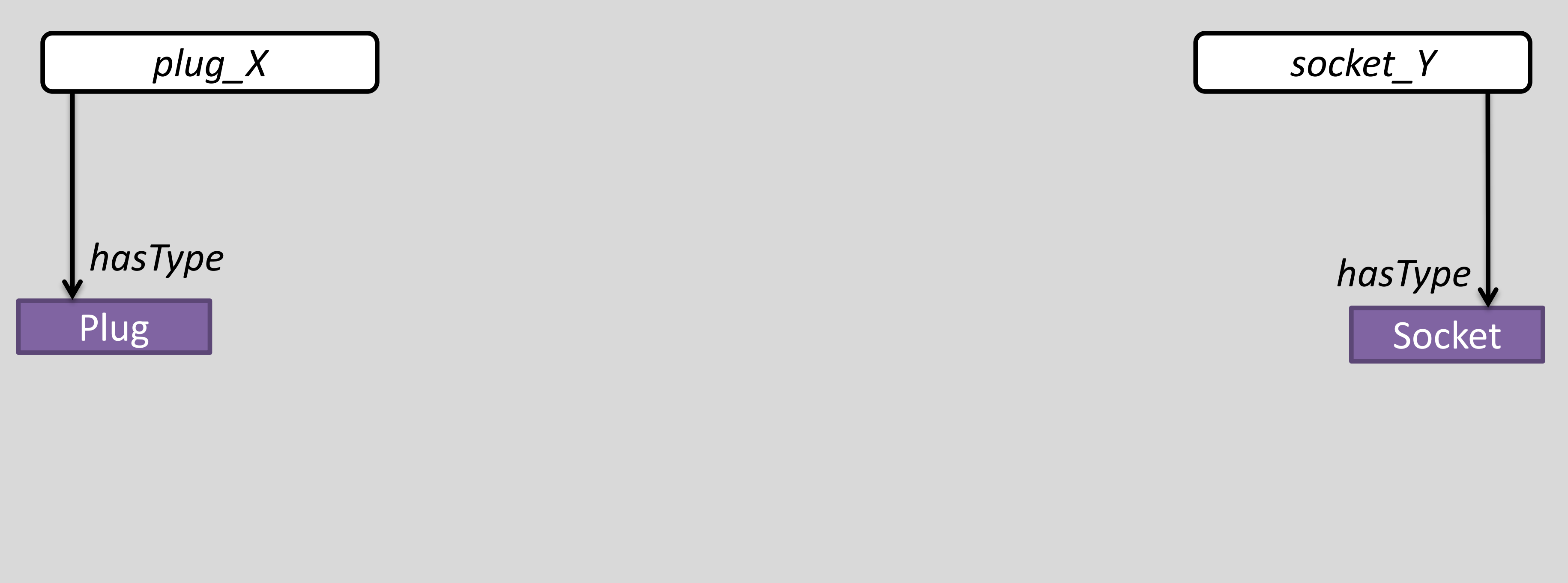
- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

What are semantic models?

- Models that describe
  - pieces of information (data, descriptions)
  - their relations → meaning (semantics)

What are semantic models?

- Models that describe
  - pieces of information (data, descriptions)
  - their relations → **meaning (semantics)**

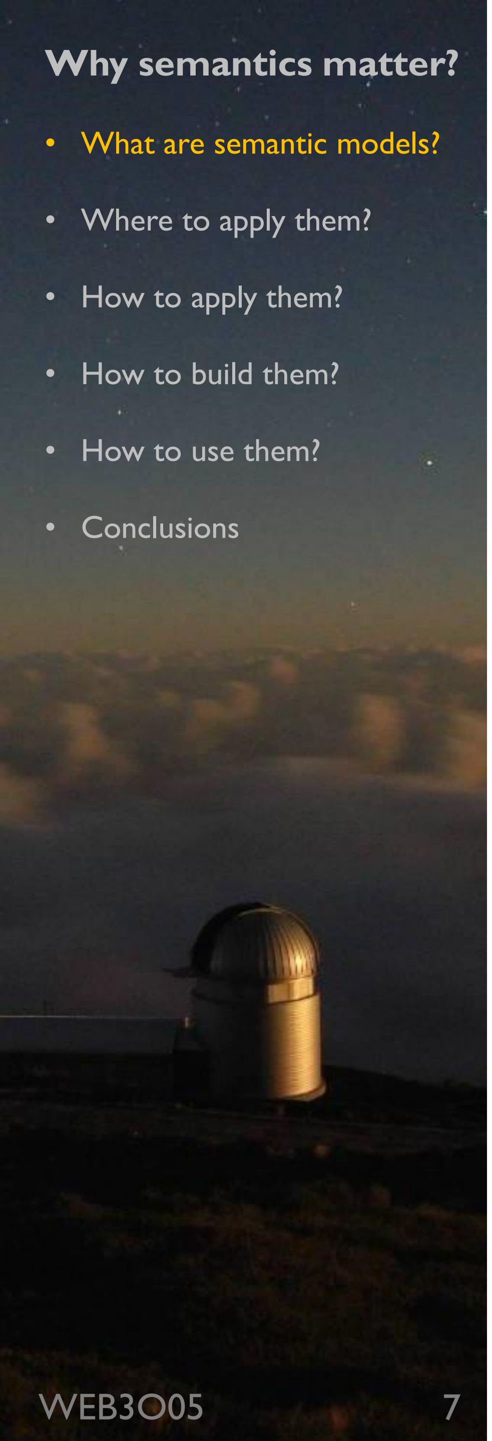


GENERIC

hasType

ELECTRIC

Plug
Socket

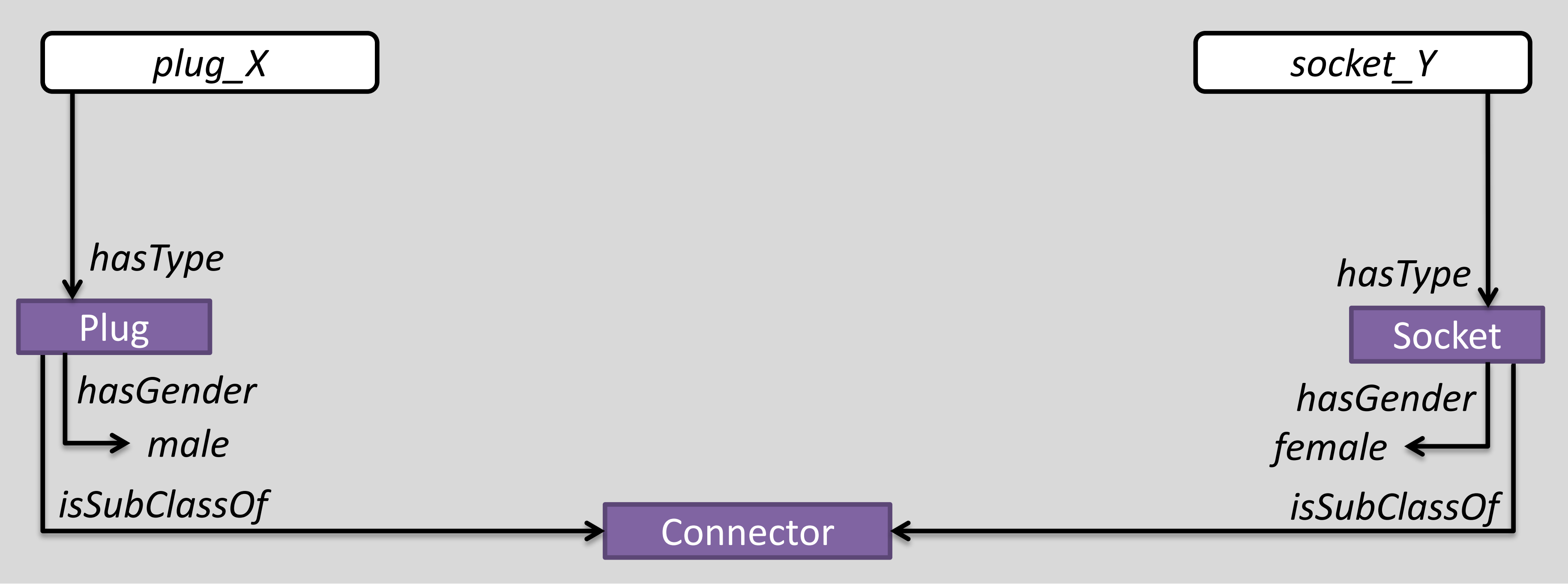


Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

What are semantic models?

- Models that describe
  - pieces of information (data, descriptions)
  - their relations → **meaning (semantics)**



GENERIC

hasType

ELECTRIC

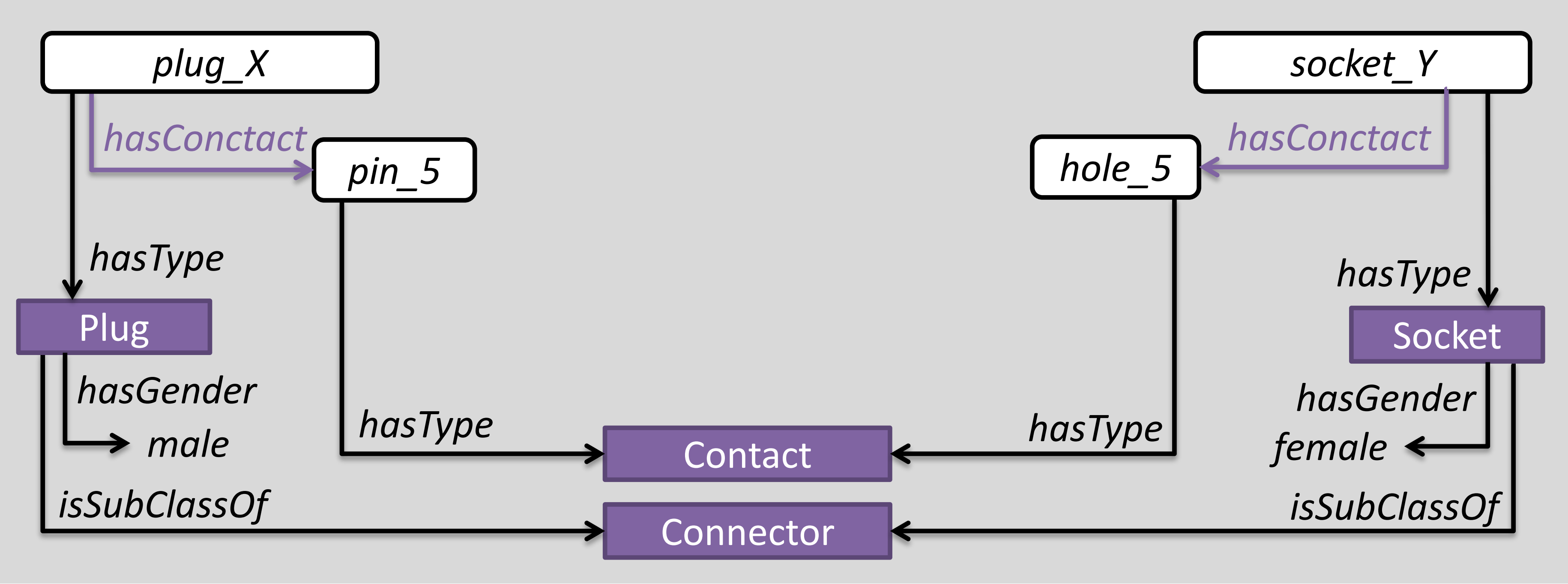
Plug
Socket

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

What are semantic models?

- Models that describe
  - pieces of information (data, descriptions)
  - their relations → meaning (semantics)

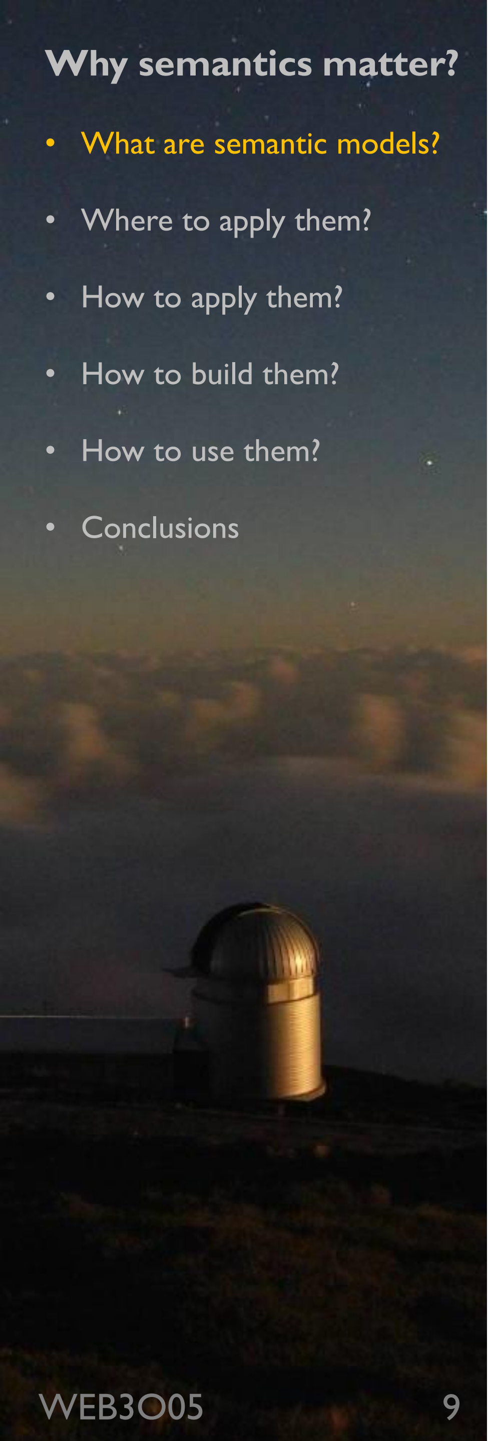


GENERIC

hasType *isSubClassOf*
hasGender

ELECTRIC

Plug Connector
Socket *hasContact*

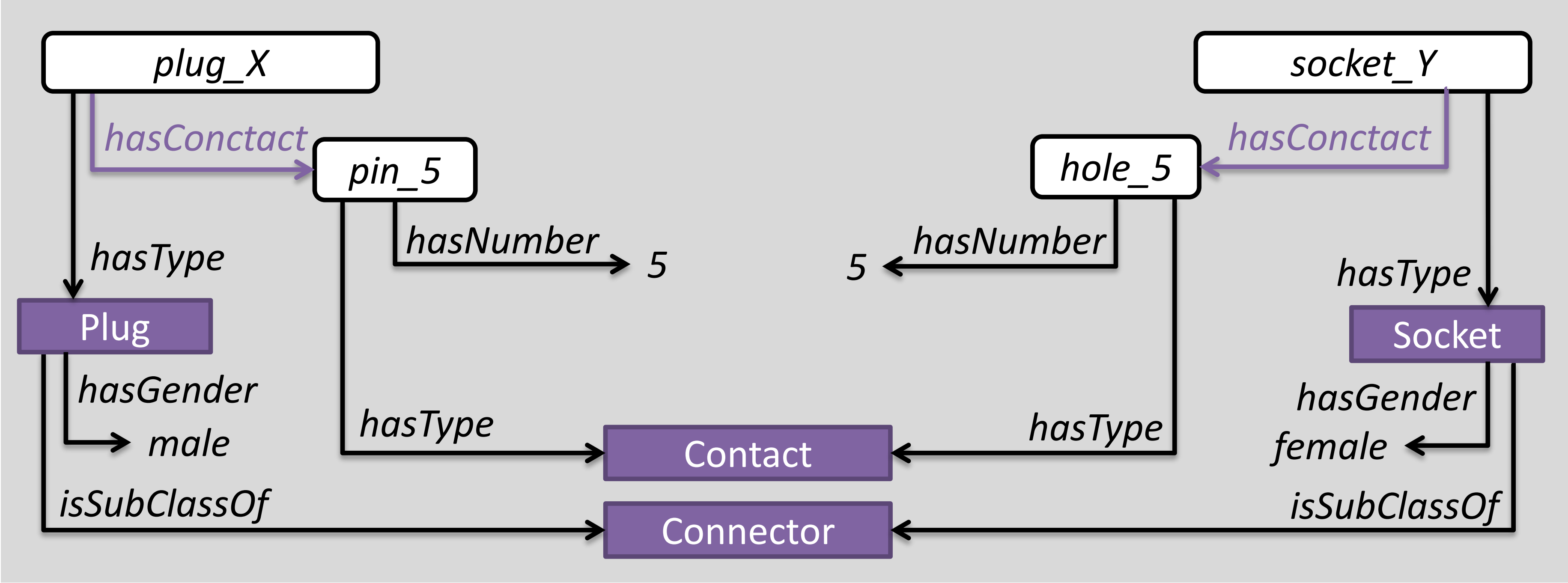


Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

What are semantic models?

- Models that describe
  - pieces of information (data, descriptions)
  - their relations ➡ meaning (semantics)



GENERIC

hasType *isSubClassOf*
hasGender *hasNumber*

ELECTRIC

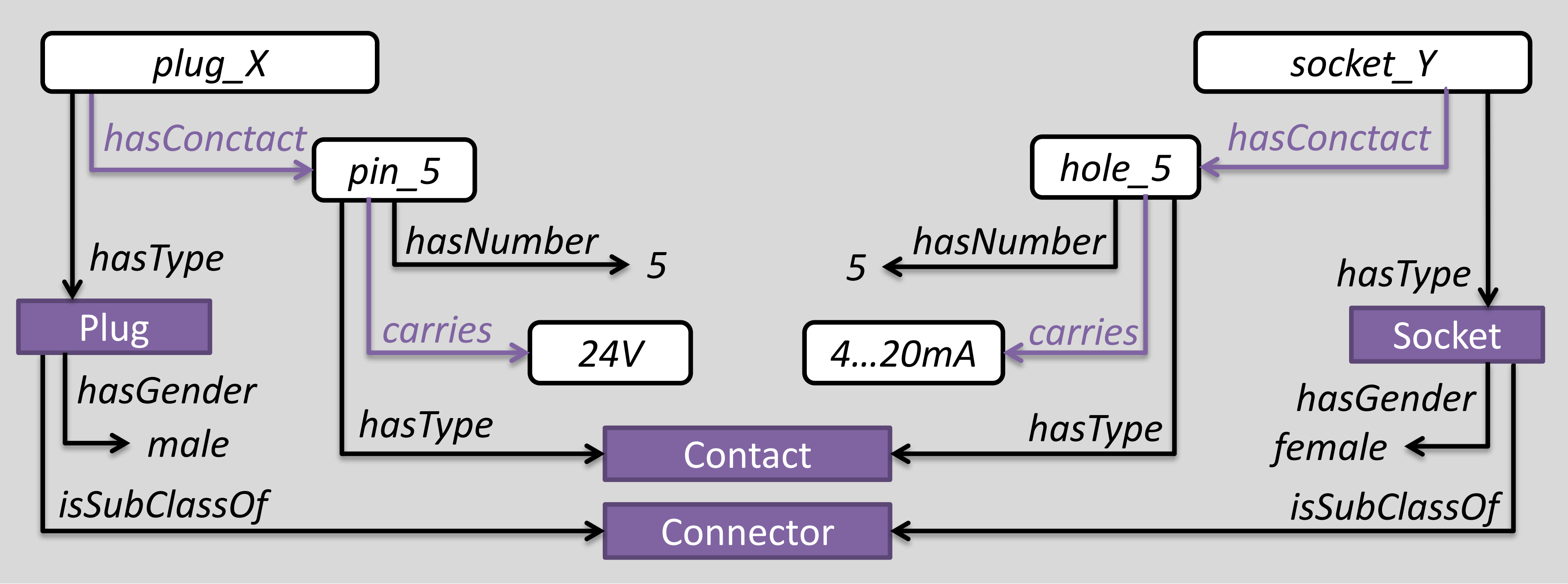
Plug Connector Contact
Socket *hasContact*

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

What are semantic models?

- Models that describe
  - pieces of information (data, descriptions)
  - their relations → meaning (semantics)



GENERIC

hasType *isSubClassOf*
hasGender *hasNumber*

ELECTRIC

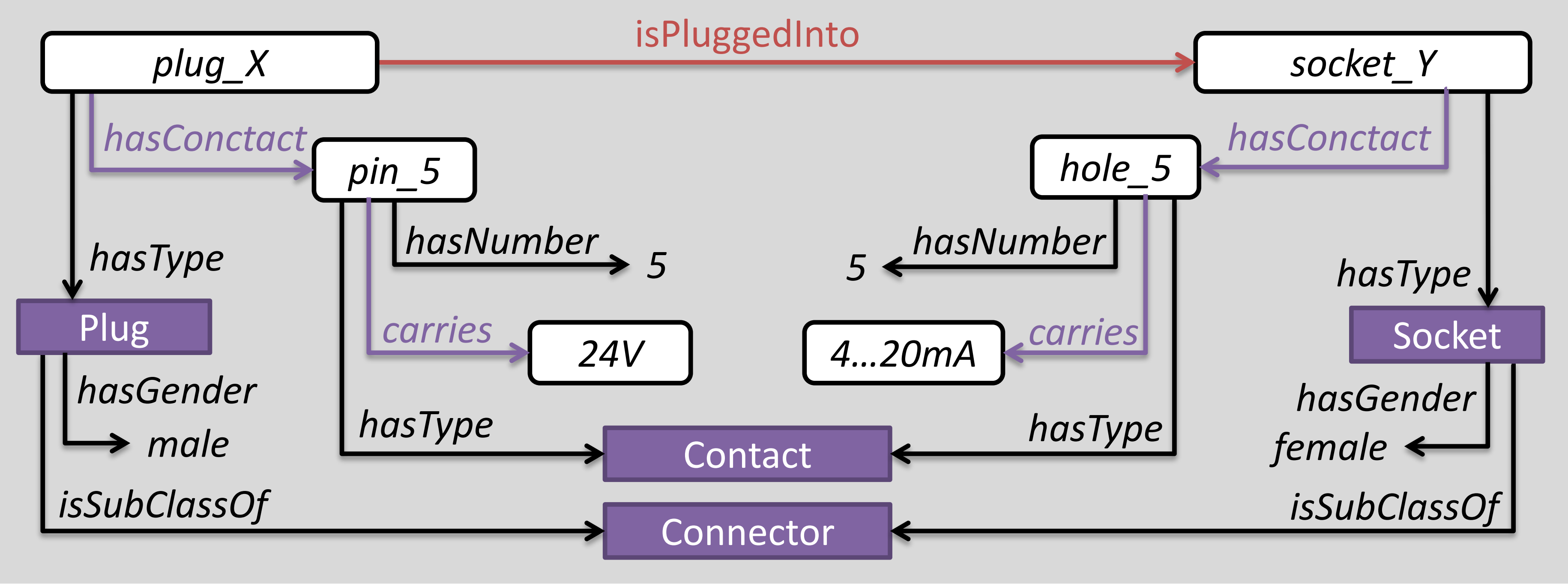
Plug Connector Contact
Socket *hasContact* *carries*

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

What are semantic models?

- Models that describe
  - pieces of information (data, descriptions)
  - their relations → meaning (semantics)



GENERIC

hasType *isSubClassOf*
hasGender *hasNumber*

ELECTRIC

Plug Connector Contact
Socket *hasContact* *carries*

MECHANIC

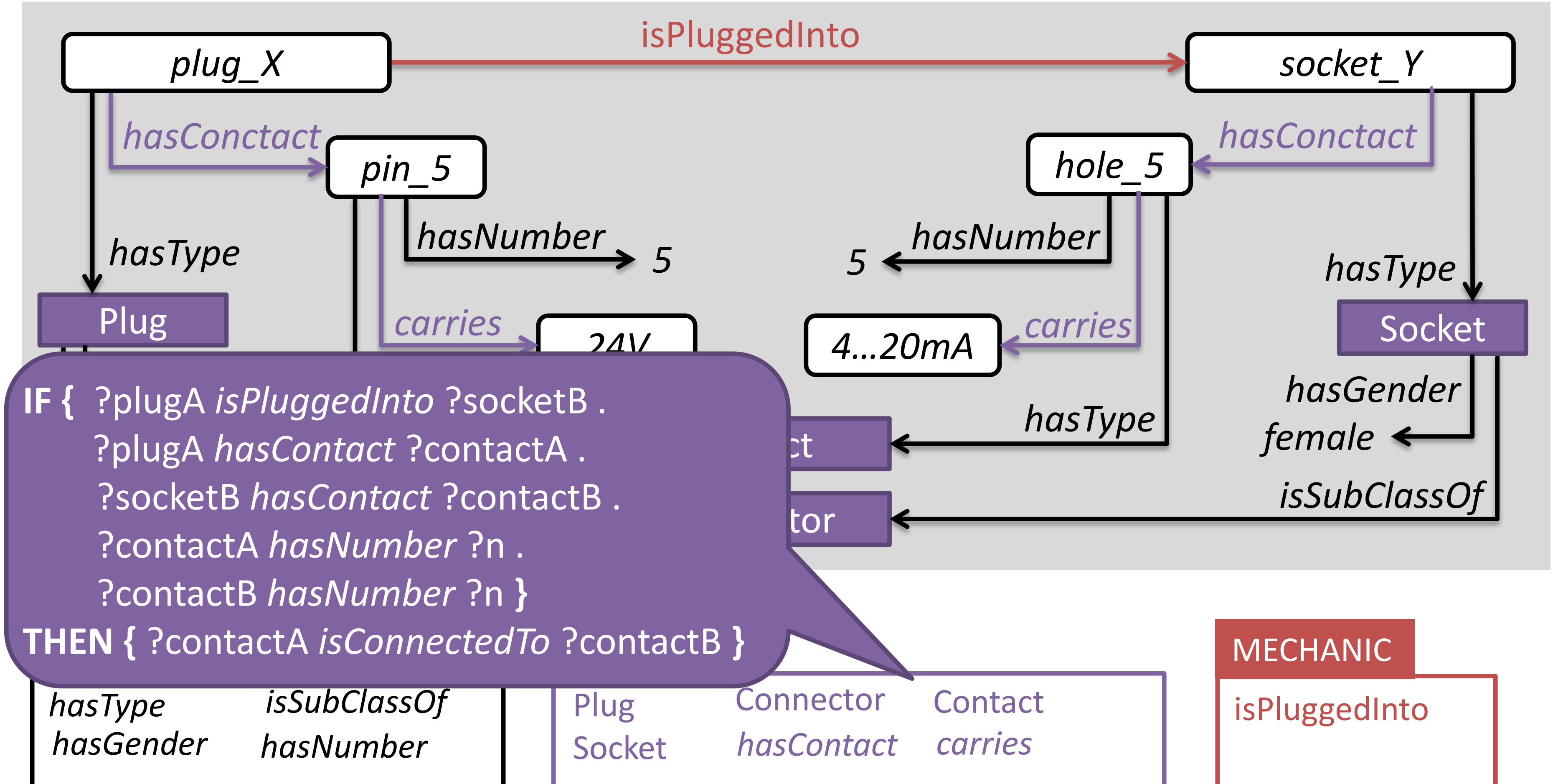
isPluggedInto

Why semantics matter?

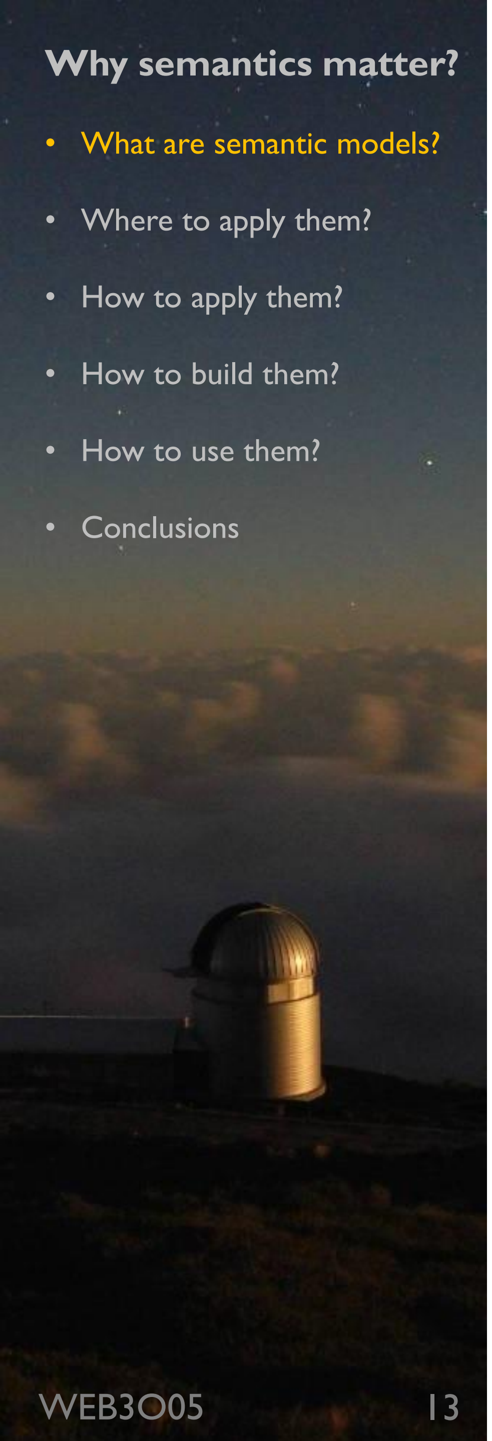
- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

What are semantic models?

- Models that describe
  - pieces of information (data, descriptions)
  - their relations → meaning (semantics)



IF { ?plugA *isPluggedInto* ?socketB .
?plugA *hasContact* ?contactA .
?socketB *hasContact* ?contactB .
?contactA *hasNumber* ?n .
?contactB *hasNumber* ?n }
THEN { ?contactA *isConnectedTo* ?contactB }

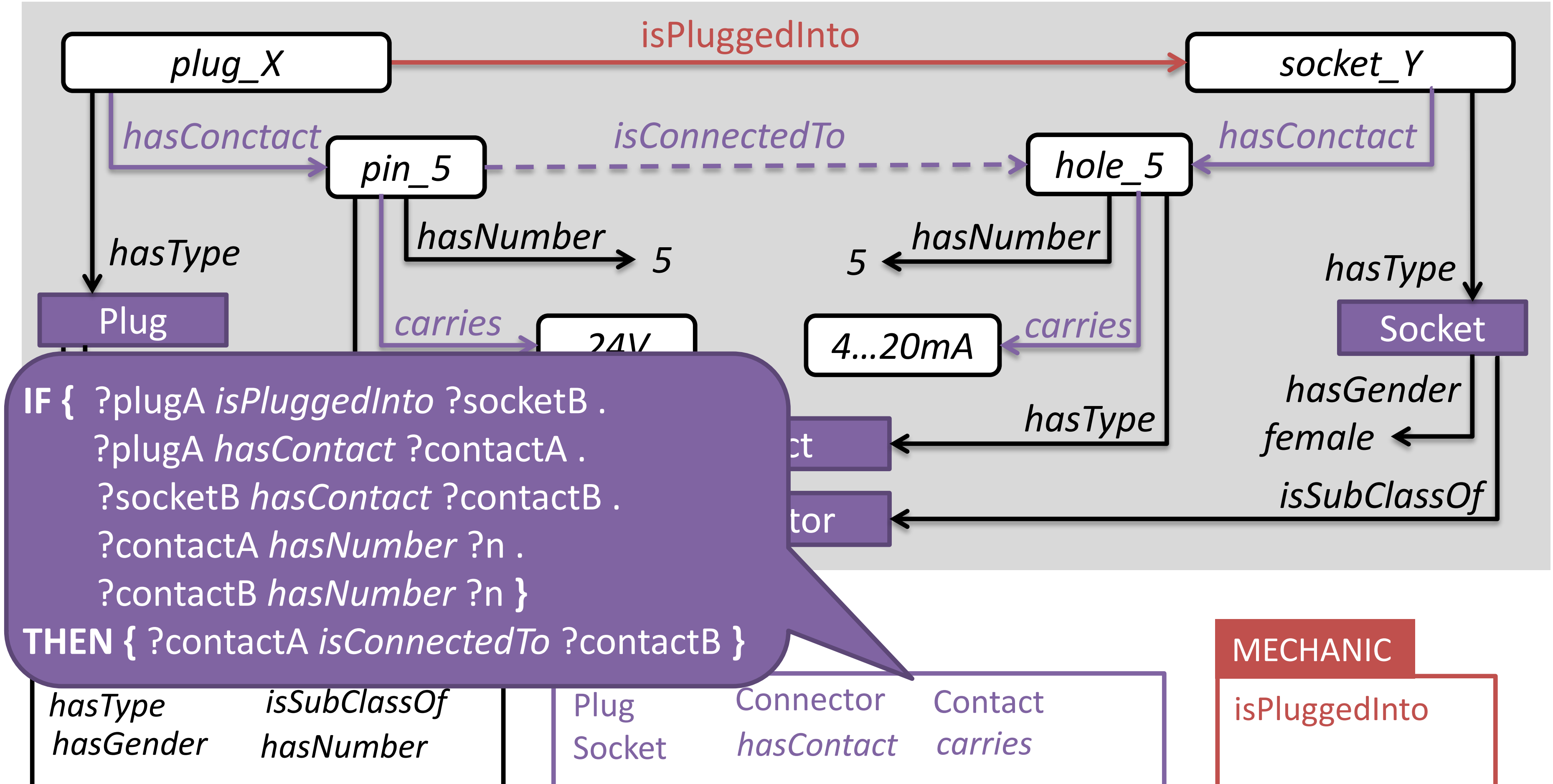


Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

What are semantic models?

- Models that describe
  - pieces of information (data, descriptions)
  - their relations → meaning (semantics)



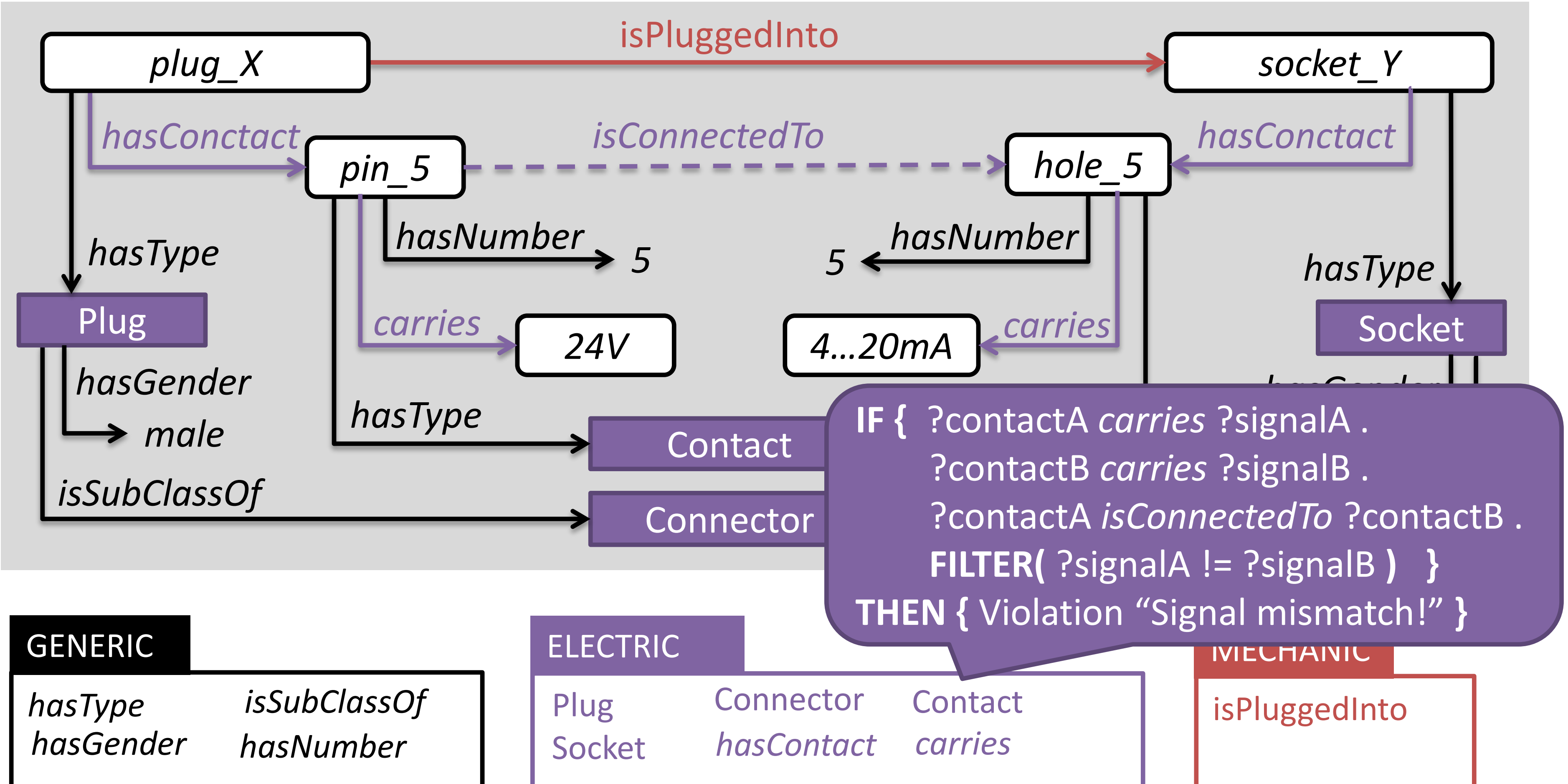
IF { ?plugA *isPluggedInto* ?socketB .
?plugA *hasContact* ?contactA .
?socketB *hasContact* ?contactB .
?contactA *hasNumber* ?n .
?contactB *hasNumber* ?n **}**
THEN { ?contactA *isConnectedTo* ?contactB **}**

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

What are semantic models?

- Models that describe
  - pieces of information (data, descriptions)
  - their relations → **meaning (semantics)**



plug_X isPluggedInto socket_Y

plug_X hasConctact pin_5

socket_Y hasConctact hole_5

pin_5 isConnectedTo hole_5

pin_5 hasNumber 5

hole_5 hasNumber 5

pin_5 carries 24V

hole_5 carries 4…20mA

plug_X hasType Plug

socket_Y hasType Socket

Plug hasGender male

Socket hasGender

pin_5 hasType Contact

Plug isSubClassOf Connector

```
IF {  ?contactA carries ?signalA .
      ?contactB carries ?signalB .
      ?contactA isConnectedTo ?contactB .
      FILTER( ?signalA != ?signalB )   }
THEN { Violation “Signal mismatch!” }
```

GENERIC: *hasType*, *hasGender*, *isSubClassOf*, *hasNumber*

ELECTRIC: Plug, Socket, Connector, *hasContact*, Contact, *carries*

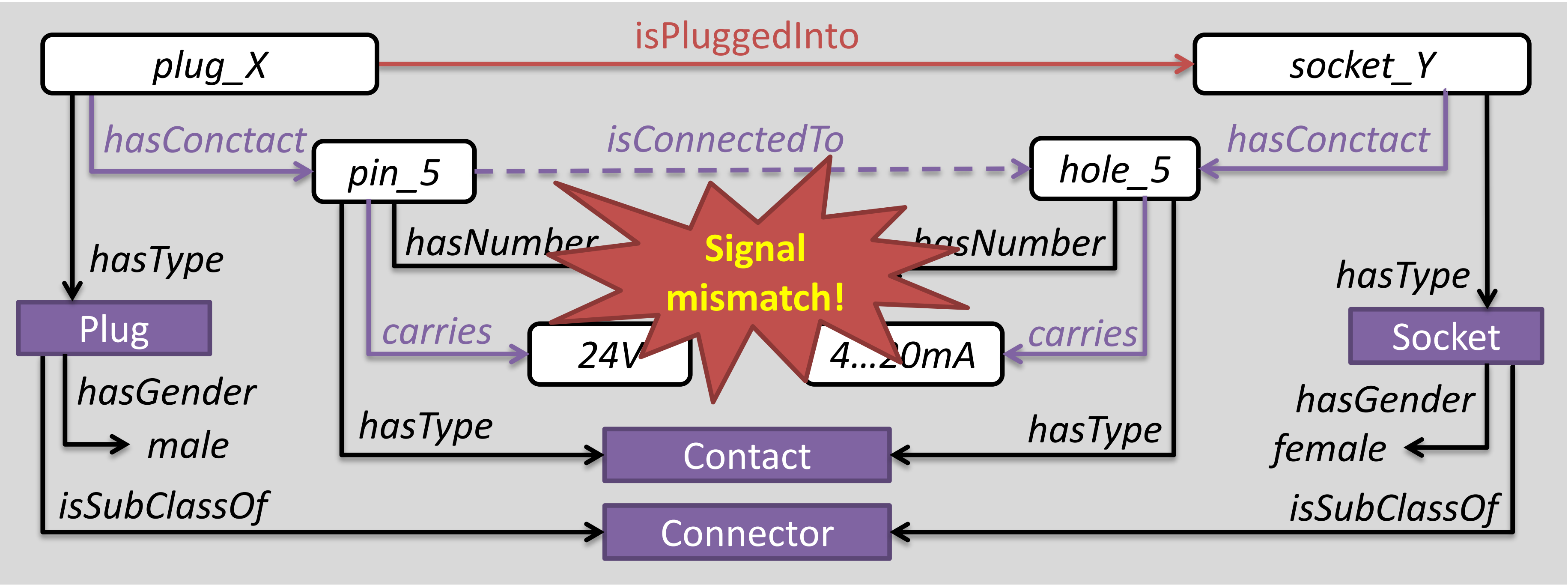
MECHANIC: isPluggedInto

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

What are semantic models?

- Models that describe
  - pieces of information (data, descriptions)
  - their relations → meaning (semantics)



GENERIC

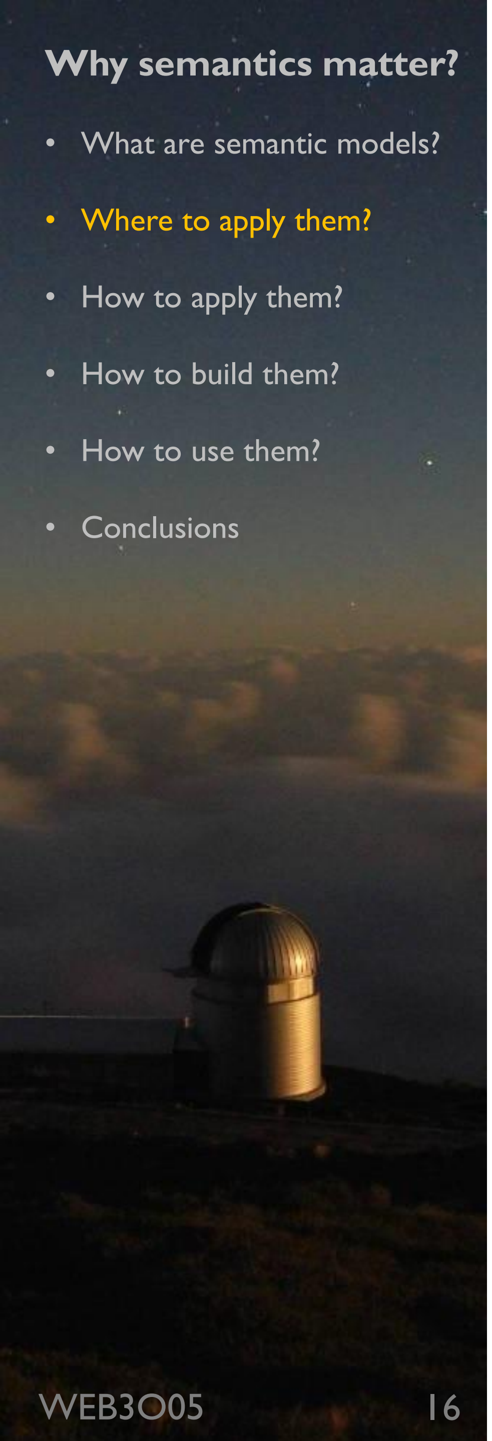
hasType *isSubClassOf*
hasGender *hasNumber*

ELECTRIC

Plug Connector Contact
Socket *hasContact* *carries*

MECHANIC

isPluggedInto



Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Where to apply them?

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Where to apply them?

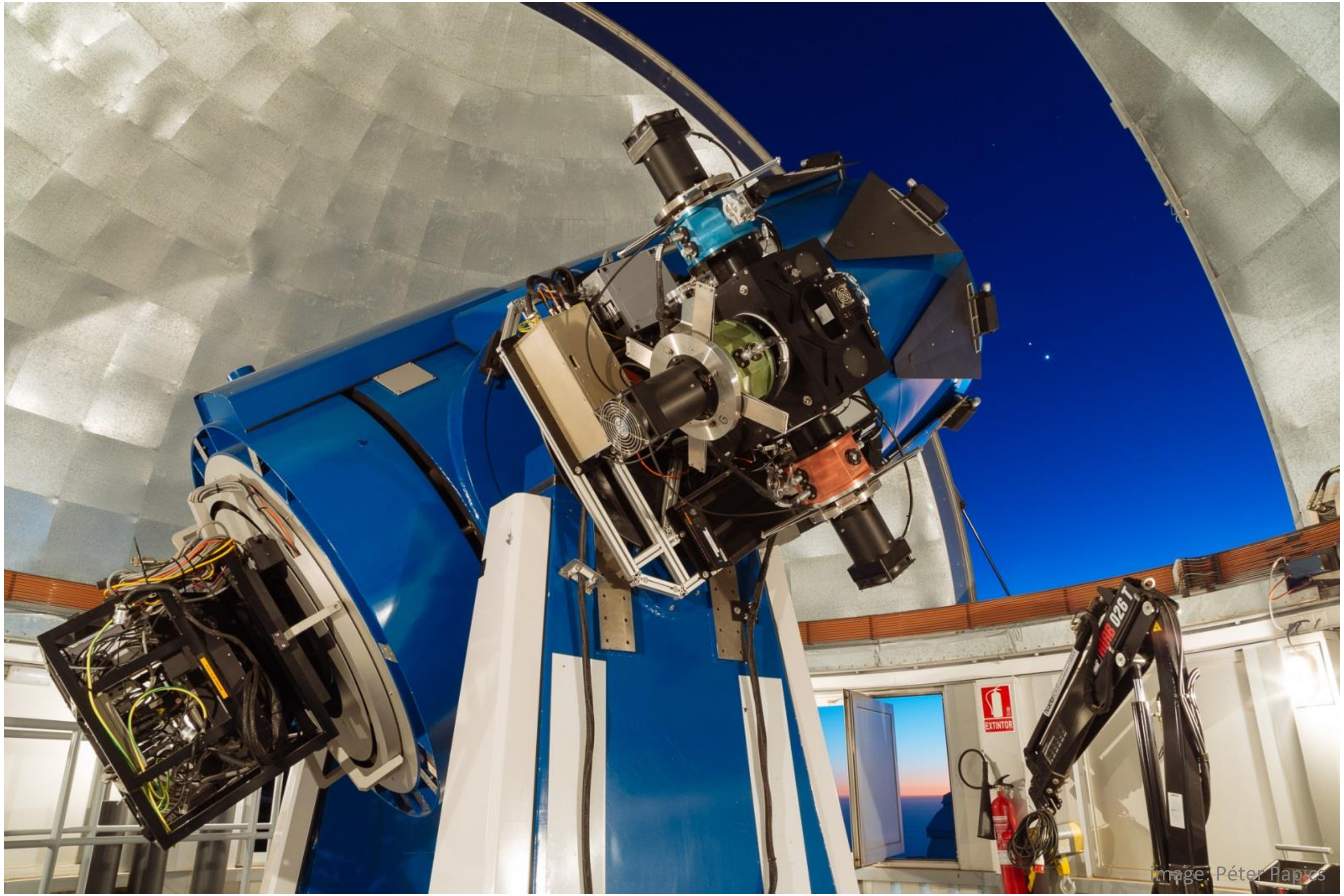
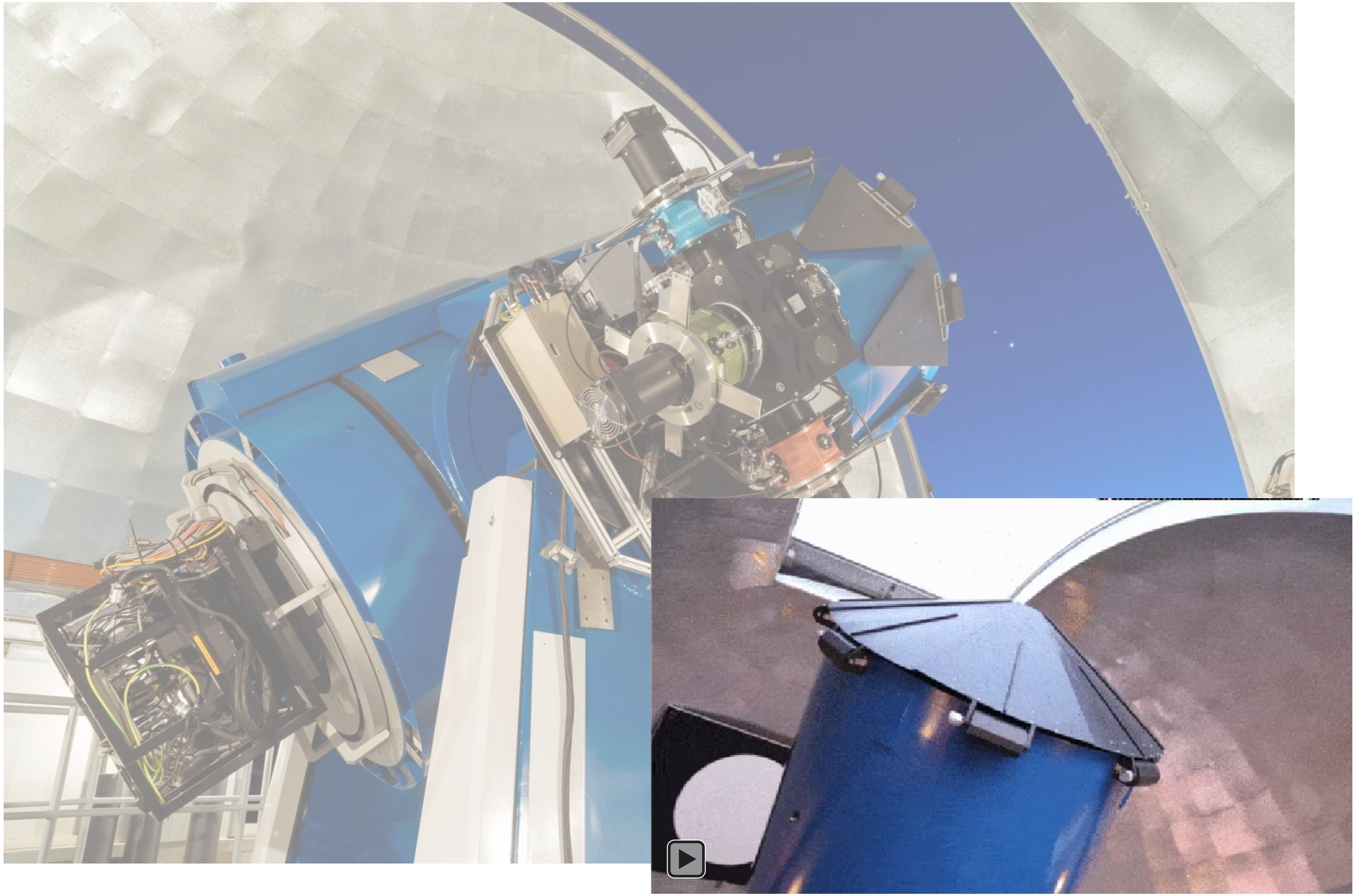


image: Péter Papics

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Where to apply them?



Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

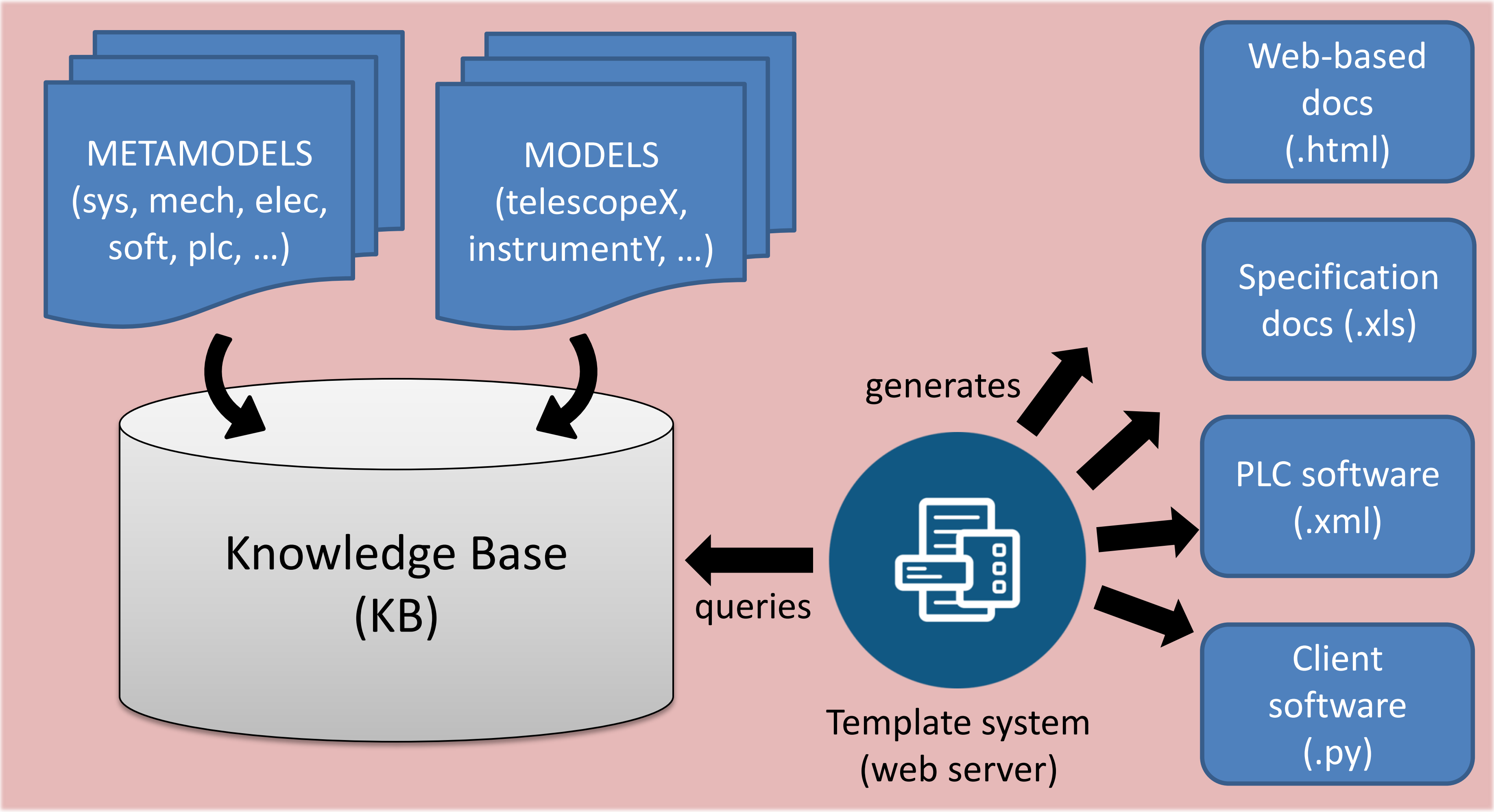
How to apply them?

Why semantics matter?

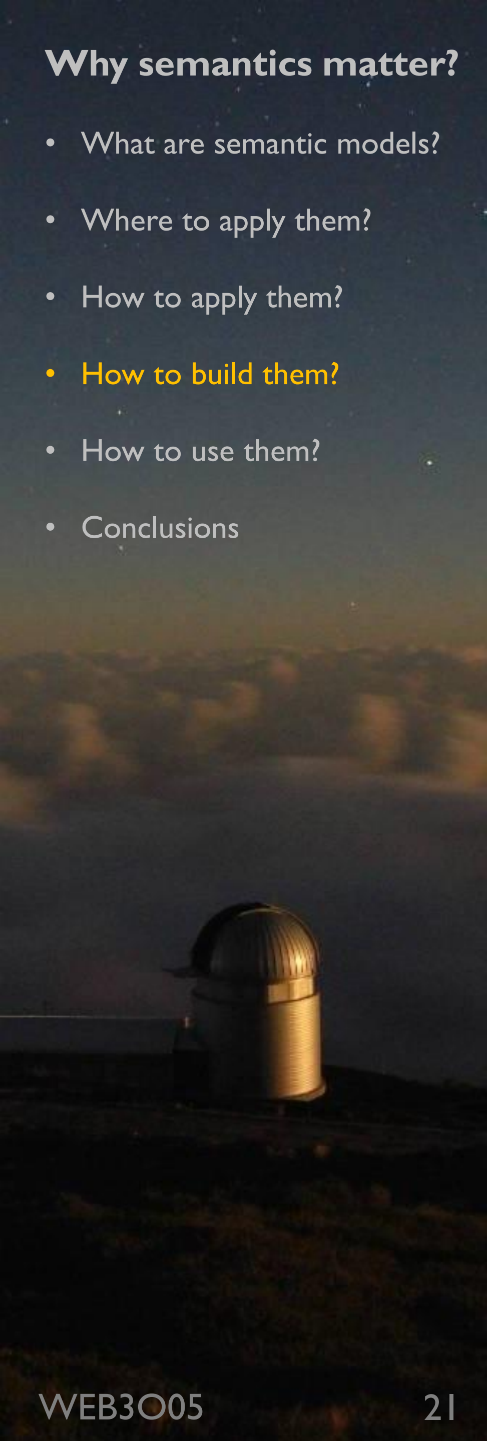
- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

How to apply them?

- Put them in a Knowledge Base and extract information!


METAMODELS (sys, mech, elec, soft, plc, …)
MODELS (telescopeX, instrumentY, …)
Knowledge Base (KB)
queries
generates
Template system (web server)
Web-based docs (.html)
Specification docs (.xls)
PLC software (.xml)
Client software (.py)

OntoManager



Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

How to build them?

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

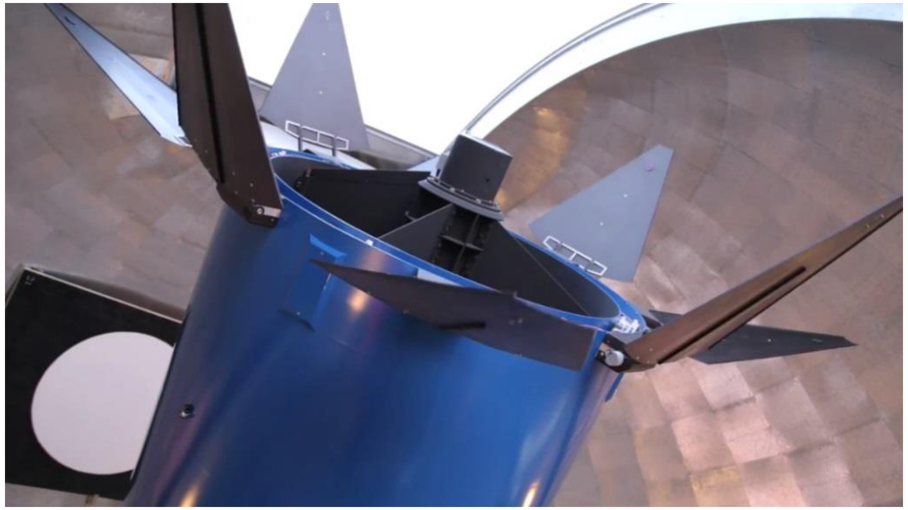
How to build them?

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

How to build them?

- Using an existing modeling language?
  - UML, SysML, … : semantics not sufficiently formal
  - Modeling languages have no “programming” capabilities (loops, functions, if-then, …)

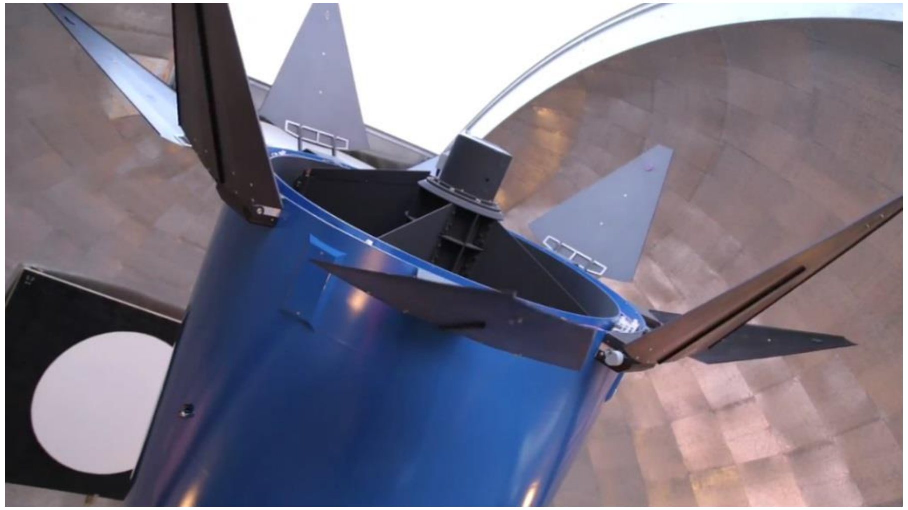


Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

How to build them?

- Using an existing modeling language?
  - UML, SysML, … : semantics not sufficiently formal
  - Modeling languages have no “programming” capabilities (loops, functions, if-then, …)



- Using a Domain Specific Language (DSL)?
  - Internal DSL called Ontoscript
  - Based on coffeescript (~javascript)
  - Idea “adopted” from the Giant Magellan Telescope project [1]

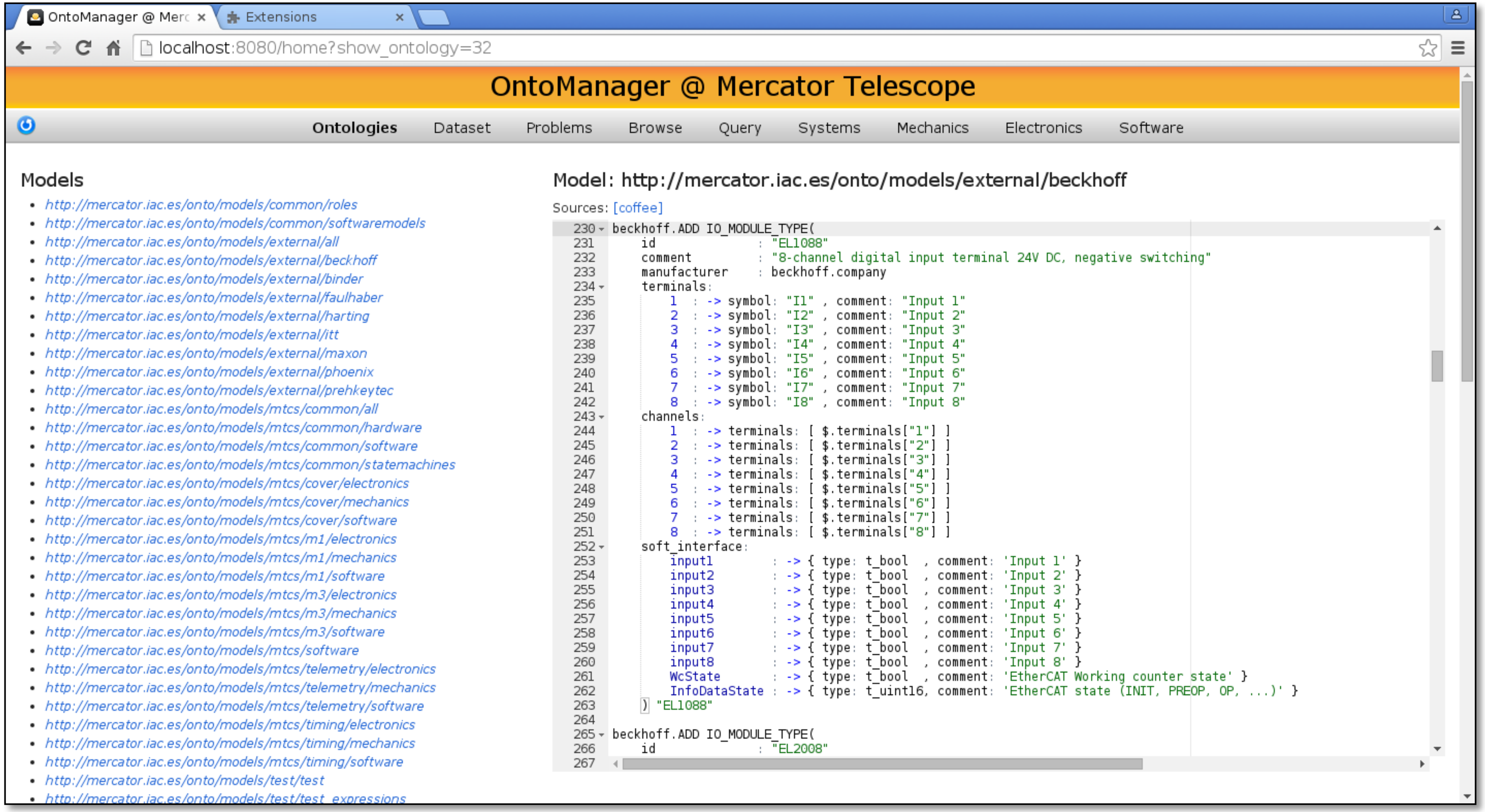
[1] J. M. Filgueira, “GMT software and controls overview”, Proc. SPIE 8451, Amsterdam, July 2012, 845111

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

How to build them?

- Example: model of an I/O module **type**



OntoManager @ Mercator Telescope

Models

- http://mercator.iac.es/onto/models/common/roles
- http://mercator.iac.es/onto/models/common/softwaremodels
- http://mercator.iac.es/onto/models/external/all
- http://mercator.iac.es/onto/models/external/beckhoff
- http://mercator.iac.es/onto/models/external/binder
- http://mercator.iac.es/onto/models/external/faulhaber
- http://mercator.iac.es/onto/models/external/harting
- http://mercator.iac.es/onto/models/external/itt
- http://mercator.iac.es/onto/models/external/maxon
- http://mercator.iac.es/onto/models/external/phoenix
- http://mercator.iac.es/onto/models/external/prehkeytec
- http://mercator.iac.es/onto/models/mtcs/common/all
- http://mercator.iac.es/onto/models/mtcs/common/hardware
- http://mercator.iac.es/onto/models/mtcs/common/software
- http://mercator.iac.es/onto/models/mtcs/common/statemachines
- http://mercator.iac.es/onto/models/mtcs/cover/electronics
- http://mercator.iac.es/onto/models/mtcs/cover/mechanics
- http://mercator.iac.es/onto/models/mtcs/cover/software
- http://mercator.iac.es/onto/models/mtcs/m1/electronics
- http://mercator.iac.es/onto/models/mtcs/m1/mechanics
- http://mercator.iac.es/onto/models/mtcs/m1/software
- http://mercator.iac.es/onto/models/mtcs/m3/electronics
- http://mercator.iac.es/onto/models/mtcs/m3/mechanics
- http://mercator.iac.es/onto/models/mtcs/m3/software
- http://mercator.iac.es/onto/models/mtcs/software
- http://mercator.iac.es/onto/models/mtcs/telemetry/electronics
- http://mercator.iac.es/onto/models/mtcs/telemetry/mechanics
- http://mercator.iac.es/onto/models/mtcs/telemetry/software
- http://mercator.iac.es/onto/models/mtcs/timing/electronics
- http://mercator.iac.es/onto/models/mtcs/timing/mechanics
- http://mercator.iac.es/onto/models/mtcs/timing/software
- http://mercator.iac.es/onto/models/test/test
- http://mercator.iac.es/onto/models/test/test_expressions

Model: http://mercator.iac.es/onto/models/external/beckhoff

Sources: [coffee]

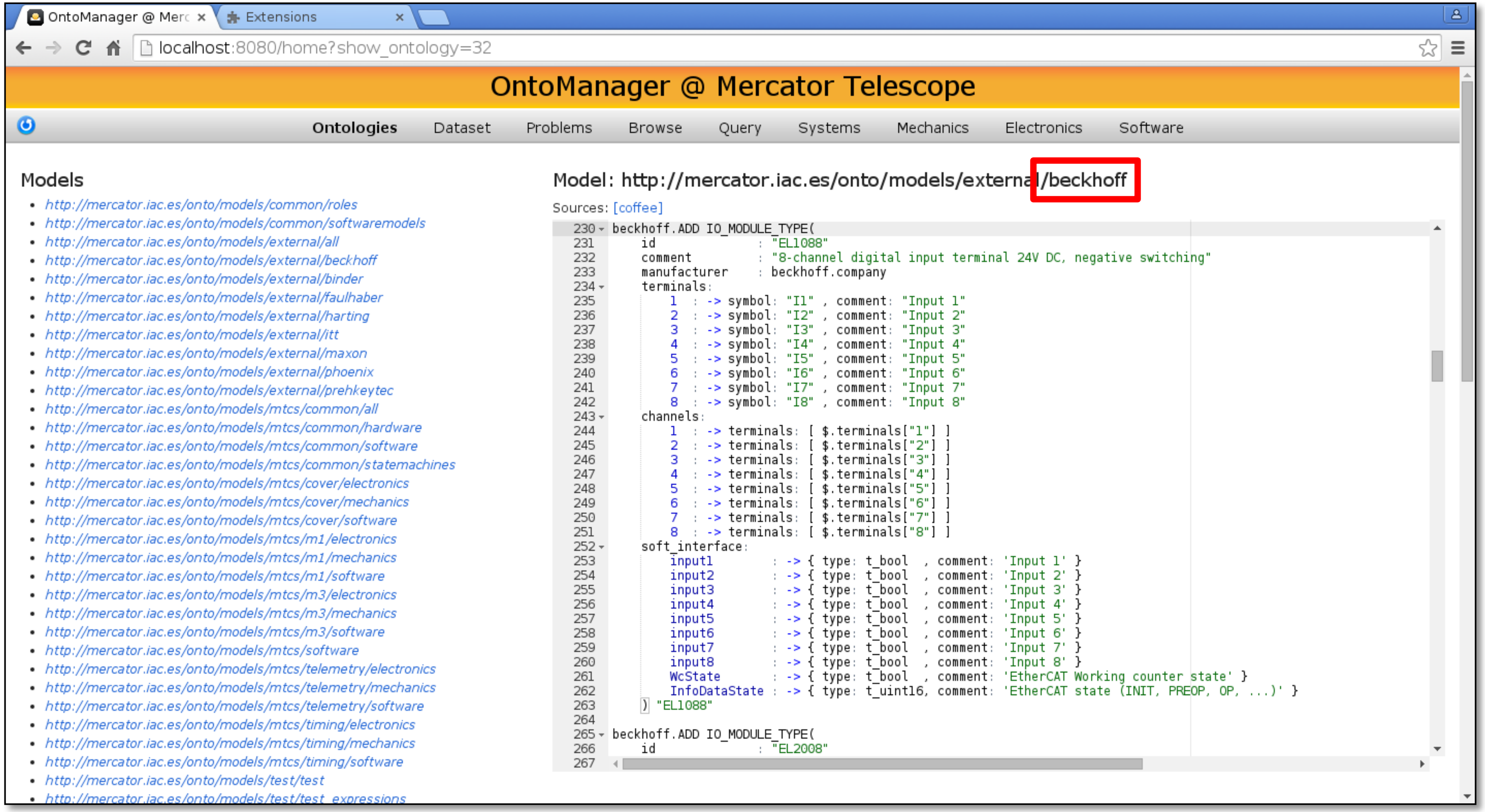
```
beckhoff.ADD IO_MODULE_TYPE(
    id              : "EL1088"
    comment         : "8-channel digital input terminal 24V DC, negative switching"
    manufacturer    : beckhoff.company
    terminals:
        1  : -> symbol: "I1" , comment: "Input 1"
        2  : -> symbol: "I2" , comment: "Input 2"
        3  : -> symbol: "I3" , comment: "Input 3"
        4  : -> symbol: "I4" , comment: "Input 4"
        5  : -> symbol: "I5" , comment: "Input 5"
        6  : -> symbol: "I6" , comment: "Input 6"
        7  : -> symbol: "I7" , comment: "Input 7"
        8  : -> symbol: "I8" , comment: "Input 8"
    channels:
        1  : -> terminals: [ $.terminals["1"] ]
        2  : -> terminals: [ $.terminals["2"] ]
        3  : -> terminals: [ $.terminals["3"] ]
        4  : -> terminals: [ $.terminals["4"] ]
        5  : -> terminals: [ $.terminals["5"] ]
        6  : -> terminals: [ $.terminals["6"] ]
        7  : -> terminals: [ $.terminals["7"] ]
        8  : -> terminals: [ $.terminals["8"] ]
    soft_interface:
        input1        : -> { type: t_bool  , comment: 'Input 1' }
        input2        : -> { type: t_bool  , comment: 'Input 2' }
        input3        : -> { type: t_bool  , comment: 'Input 3' }
        input4        : -> { type: t_bool  , comment: 'Input 4' }
        input5        : -> { type: t_bool  , comment: 'Input 5' }
        input6        : -> { type: t_bool  , comment: 'Input 6' }
        input7        : -> { type: t_bool  , comment: 'Input 7' }
        input8        : -> { type: t_bool  , comment: 'Input 8' }
        WcState       : -> { type: t_bool  , comment: 'EtherCAT Working counter state' }
        InfoDataState : -> { type: t_uint16, comment: 'EtherCAT state (INIT, PREOP, OP, ...)' }
    ) "EL1088"

beckhoff.ADD IO_MODULE_TYPE(
    id              : "EL2008"
```

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

How to build them?

- Example: model of an I/O module **type**



Models

- http://mercator.iac.es/onto/models/common/roles
- http://mercator.iac.es/onto/models/common/softwaremodels
- http://mercator.iac.es/onto/models/external/all
- http://mercator.iac.es/onto/models/external/beckhoff
- http://mercator.iac.es/onto/models/external/binder
- http://mercator.iac.es/onto/models/external/faulhaber
- http://mercator.iac.es/onto/models/external/harting
- http://mercator.iac.es/onto/models/external/itt
- http://mercator.iac.es/onto/models/external/maxon
- http://mercator.iac.es/onto/models/external/phoenix
- http://mercator.iac.es/onto/models/external/prehkeytec
- http://mercator.iac.es/onto/models/mtcs/common/all
- http://mercator.iac.es/onto/models/mtcs/common/hardware
- http://mercator.iac.es/onto/models/mtcs/common/software
- http://mercator.iac.es/onto/models/mtcs/common/statemachines
- http://mercator.iac.es/onto/models/mtcs/cover/electronics
- http://mercator.iac.es/onto/models/mtcs/cover/mechanics
- http://mercator.iac.es/onto/models/mtcs/cover/software
- http://mercator.iac.es/onto/models/mtcs/m1/electronics
- http://mercator.iac.es/onto/models/mtcs/m1/mechanics
- http://mercator.iac.es/onto/models/mtcs/m1/software
- http://mercator.iac.es/onto/models/mtcs/m3/electronics
- http://mercator.iac.es/onto/models/mtcs/m3/mechanics
- http://mercator.iac.es/onto/models/mtcs/m3/software
- http://mercator.iac.es/onto/models/mtcs/software
- http://mercator.iac.es/onto/models/mtcs/telemetry/electronics
- http://mercator.iac.es/onto/models/mtcs/telemetry/mechanics
- http://mercator.iac.es/onto/models/mtcs/telemetry/software
- http://mercator.iac.es/onto/models/mtcs/timing/electronics
- http://mercator.iac.es/onto/models/mtcs/timing/mechanics
- http://mercator.iac.es/onto/models/mtcs/timing/software
- http://mercator.iac.es/onto/models/test/test
- http://mercator.iac.es/onto/models/test/test_expressions

Model: http://mercator.iac.es/onto/models/external/beckhoff

Sources: [coffee]

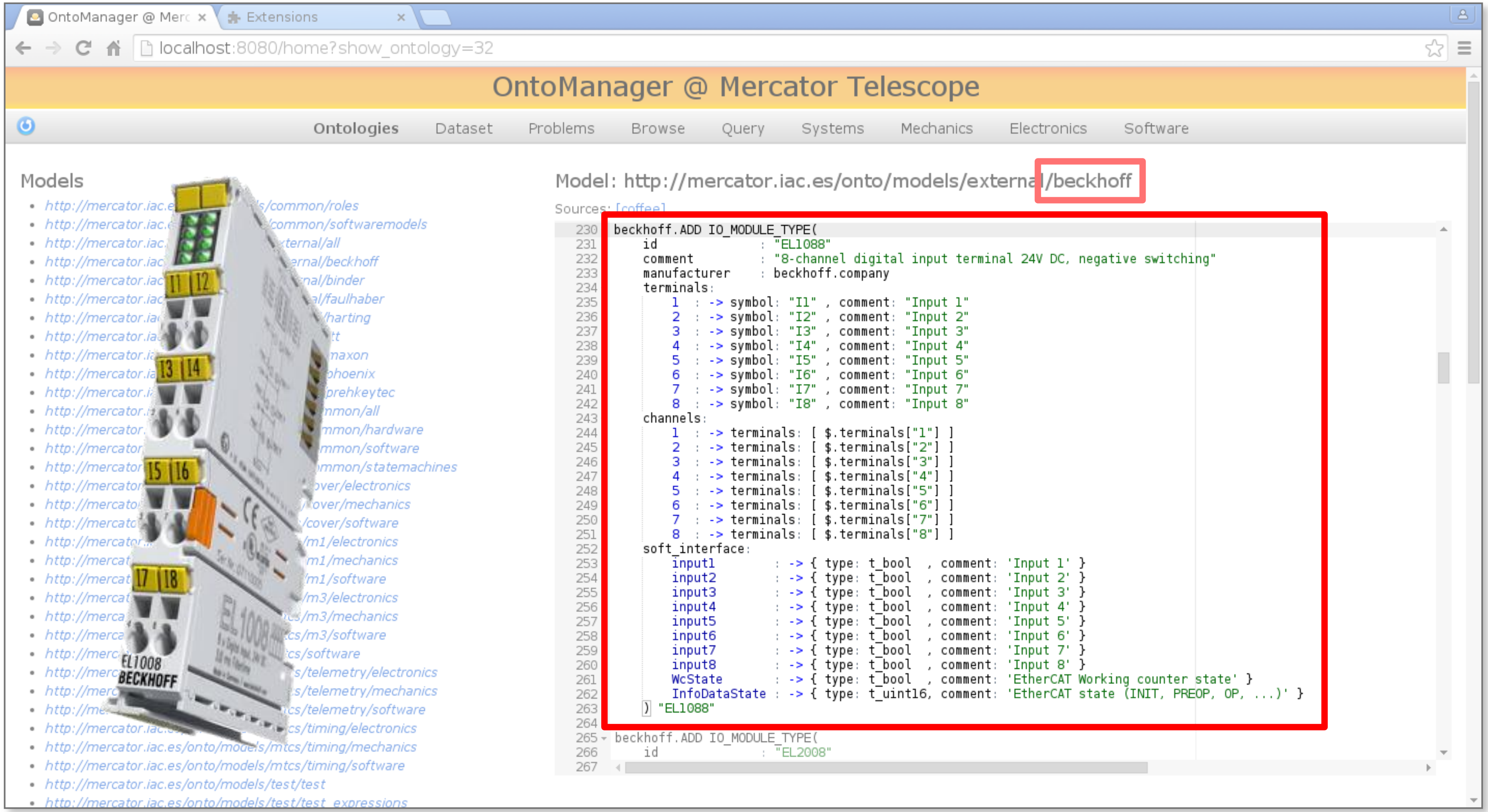
```
beckhoff.ADD IO_MODULE_TYPE(
    id                 : "EL1088"
    comment            : "8-channel digital input terminal 24V DC, negative switching"
    manufacturer       : beckhoff.company
    terminals:
        1  : -> symbol: "I1" , comment: "Input 1"
        2  : -> symbol: "I2" , comment: "Input 2"
        3  : -> symbol: "I3" , comment: "Input 3"
        4  : -> symbol: "I4" , comment: "Input 4"
        5  : -> symbol: "I5" , comment: "Input 5"
        6  : -> symbol: "I6" , comment: "Input 6"
        7  : -> symbol: "I7" , comment: "Input 7"
        8  : -> symbol: "I8" , comment: "Input 8"
    channels:
        1  : -> terminals: [ $.terminals["1"] ]
        2  : -> terminals: [ $.terminals["2"] ]
        3  : -> terminals: [ $.terminals["3"] ]
        4  : -> terminals: [ $.terminals["4"] ]
        5  : -> terminals: [ $.terminals["5"] ]
        6  : -> terminals: [ $.terminals["6"] ]
        7  : -> terminals: [ $.terminals["7"] ]
        8  : -> terminals: [ $.terminals["8"] ]
    soft_interface:
        input1        : -> { type: t_bool  , comment: 'Input 1' }
        input2        : -> { type: t_bool  , comment: 'Input 2' }
        input3        : -> { type: t_bool  , comment: 'Input 3' }
        input4        : -> { type: t_bool  , comment: 'Input 4' }
        input5        : -> { type: t_bool  , comment: 'Input 5' }
        input6        : -> { type: t_bool  , comment: 'Input 6' }
        input7        : -> { type: t_bool  , comment: 'Input 7' }
        input8        : -> { type: t_bool  , comment: 'Input 8' }
        WcState       : -> { type: t_bool  , comment: 'EtherCAT Working counter state' }
        InfoDataState : -> { type: t_uint16, comment: 'EtherCAT state (INIT, PREOP, OP, ...)' }
    ) "EL1088"

beckhoff.ADD IO_MODULE_TYPE(
    id                 : "EL2008"
```

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

How to build them?

- Example: model of an I/O module **type**



Models

- http://mercator.iac.e /common/roles
- http://mercator.iac. /common/softwaremodels
- http://mercator.iac. xternal/all
- http://mercator.iac ernal/beckhoff
- http://mercator.iac nal/binder
- http://mercator.iac al/faulhaber
- http://mercator.iac /harting
- http://mercator.ia tt
- http://mercator.ia maxon
- http://mercator.ia phoenix
- http://mercator.i prehkeytec
- http://mercator. mmon/all
- http://mercator. mmon/hardware
- http://mercator mmon/software
- http://mercator mmon/statemachines
- http://mercator over/electronics
- http://mercato over/mechanics
- http://mercato /cover/software
- http://mercator /m1/electronics
- http://mercat /m1/mechanics
- http://mercat /m1/software
- http://mercat /m3/electronics
- http://merca cs/m3/mechanics
- http://merca cs/m3/software
- http://merc cs/software
- http://merc s/telemetry/electronics
- http://merc s/telemetry/mechanics
- http://me cs/telemetry/software
- http://mercator.iac. cs/timing/electronics
- http://mercator.iac.es/onto/models/mtcs/timing/mechanics
- http://mercator.iac.es/onto/models/mtcs/timing/software
- http://mercator.iac.es/onto/models/test/test
- http://mercator.iac.es/onto/models/test/test_expressions

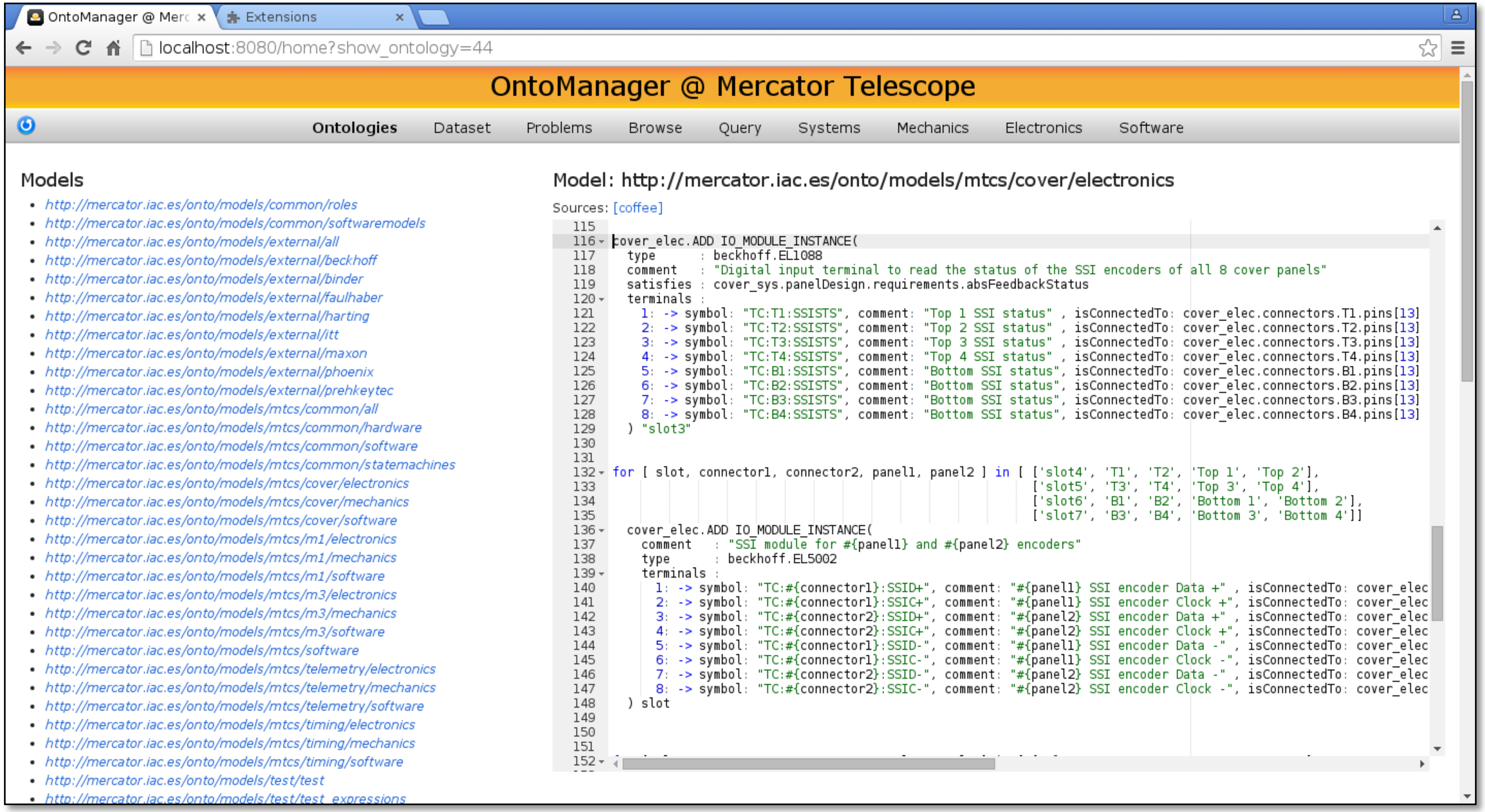
Model: http://mercator.iac.es/onto/models/external/beckhoff

Sources: [coffee]

```
beckhoff.ADD IO_MODULE_TYPE(
    id                : "EL1088"
    comment           : "8-channel digital input terminal 24V DC, negative switching"
    manufacturer      : beckhoff.company
    terminals:
        1  : -> symbol: "I1" , comment: "Input 1"
        2  : -> symbol: "I2" , comment: "Input 2"
        3  : -> symbol: "I3" , comment: "Input 3"
        4  : -> symbol: "I4" , comment: "Input 4"
        5  : -> symbol: "I5" , comment: "Input 5"
        6  : -> symbol: "I6" , comment: "Input 6"
        7  : -> symbol: "I7" , comment: "Input 7"
        8  : -> symbol: "I8" , comment: "Input 8"
    channels:
        1  : -> terminals: [ $.terminals["1"] ]
        2  : -> terminals: [ $.terminals["2"] ]
        3  : -> terminals: [ $.terminals["3"] ]
        4  : -> terminals: [ $.terminals["4"] ]
        5  : -> terminals: [ $.terminals["5"] ]
        6  : -> terminals: [ $.terminals["6"] ]
        7  : -> terminals: [ $.terminals["7"] ]
        8  : -> terminals: [ $.terminals["8"] ]
    soft_interface:
        input1        : -> { type: t_bool  , comment: 'Input 1' }
        input2        : -> { type: t_bool  , comment: 'Input 2' }
        input3        : -> { type: t_bool  , comment: 'Input 3' }
        input4        : -> { type: t_bool  , comment: 'Input 4' }
        input5        : -> { type: t_bool  , comment: 'Input 5' }
        input6        : -> { type: t_bool  , comment: 'Input 6' }
        input7        : -> { type: t_bool  , comment: 'Input 7' }
        input8        : -> { type: t_bool  , comment: 'Input 8' }
        WcState       : -> { type: t_bool  , comment: 'EtherCAT Working counter state' }
        InfoDataState : -> { type: t_uint16, comment: 'EtherCAT state (INIT, PREOP, OP, ...)' }
    ) "EL1088"

beckhoff.ADD IO_MODULE_TYPE(
    id                : "EL2008"
```

How to build them?

- Example: model of an I/O module **instance**



OntoManager @ Mercator Telescope

Models

- *http://mercator.iac.es/onto/models/common/roles*
- *http://mercator.iac.es/onto/models/common/softwaremodels*
- *http://mercator.iac.es/onto/models/external/all*
- *http://mercator.iac.es/onto/models/external/beckhoff*
- *http://mercator.iac.es/onto/models/external/binder*
- *http://mercator.iac.es/onto/models/external/faulhaber*
- *http://mercator.iac.es/onto/models/external/harting*
- *http://mercator.iac.es/onto/models/external/itt*
- *http://mercator.iac.es/onto/models/external/maxon*
- *http://mercator.iac.es/onto/models/external/phoenix*
- *http://mercator.iac.es/onto/models/external/prehkeytec*
- *http://mercator.iac.es/onto/models/mtcs/common/all*
- *http://mercator.iac.es/onto/models/mtcs/common/hardware*
- *http://mercator.iac.es/onto/models/mtcs/common/software*
- *http://mercator.iac.es/onto/models/mtcs/common/statemachines*
- *http://mercator.iac.es/onto/models/mtcs/cover/electronics*
- *http://mercator.iac.es/onto/models/mtcs/cover/mechanics*
- *http://mercator.iac.es/onto/models/mtcs/cover/software*
- *http://mercator.iac.es/onto/models/mtcs/m1/electronics*
- *http://mercator.iac.es/onto/models/mtcs/m1/mechanics*
- *http://mercator.iac.es/onto/models/mtcs/m1/software*
- *http://mercator.iac.es/onto/models/mtcs/m3/electronics*
- *http://mercator.iac.es/onto/models/mtcs/m3/mechanics*
- *http://mercator.iac.es/onto/models/mtcs/m3/software*
- *http://mercator.iac.es/onto/models/mtcs/software*
- *http://mercator.iac.es/onto/models/mtcs/telemetry/electronics*
- *http://mercator.iac.es/onto/models/mtcs/telemetry/mechanics*
- *http://mercator.iac.es/onto/models/mtcs/telemetry/software*
- *http://mercator.iac.es/onto/models/mtcs/timing/electronics*
- *http://mercator.iac.es/onto/models/mtcs/timing/mechanics*
- *http://mercator.iac.es/onto/models/mtcs/timing/software*
- *http://mercator.iac.es/onto/models/test/test*
- *http://mercator.iac.es/onto/models/test/test_expressions*

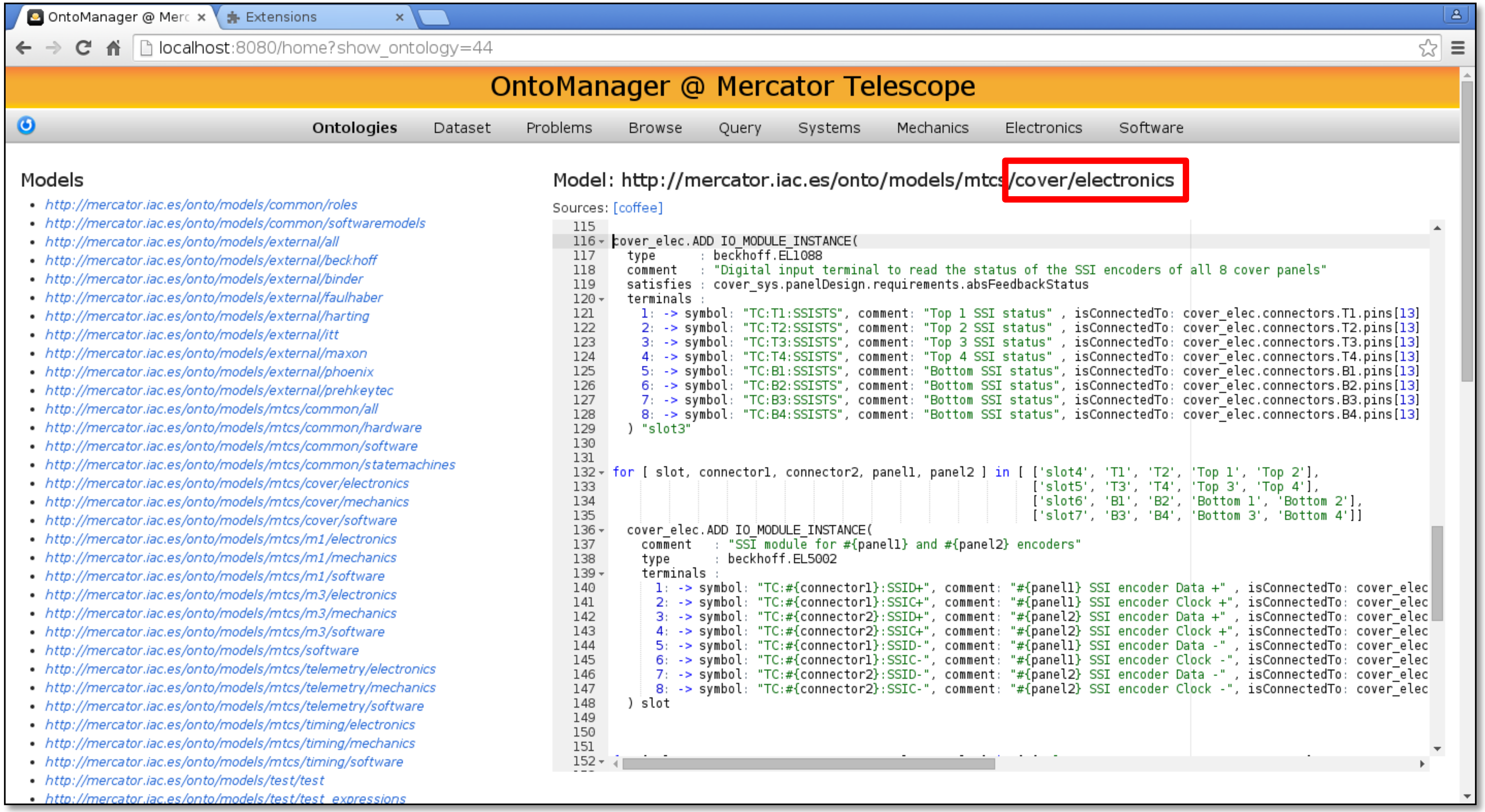
Model: http://mercator.iac.es/onto/models/mtcs/cover/electronics

Sources: [coffee]

```

cover_elec.ADD IO_MODULE_INSTANCE(
  type      : beckhoff.EL1088
  comment   : "Digital input terminal to read the status of the SSI encoders of all 8 cover panels"
  satisfies : cover_sys.panelDesign.requirements.absFeedbackStatus
  terminals :
    1: -> symbol: "TC:T1:SSISTS", comment: "Top 1 SSI status" , isConnectedTo: cover_elec.connectors.T1.pins[13]
    2: -> symbol: "TC:T2:SSISTS", comment: "Top 2 SSI status" , isConnectedTo: cover_elec.connectors.T2.pins[13]
    3: -> symbol: "TC:T3:SSISTS", comment: "Top 3 SSI status" , isConnectedTo: cover_elec.connectors.T3.pins[13]
    4: -> symbol: "TC:T4:SSISTS", comment: "Top 4 SSI status" , isConnectedTo: cover_elec.connectors.T4.pins[13]
    5: -> symbol: "TC:B1:SSISTS", comment: "Bottom SSI status", isConnectedTo: cover_elec.connectors.B1.pins[13]
    6: -> symbol: "TC:B2:SSISTS", comment: "Bottom SSI status", isConnectedTo: cover_elec.connectors.B2.pins[13]
    7: -> symbol: "TC:B3:SSISTS", comment: "Bottom SSI status", isConnectedTo: cover_elec.connectors.B3.pins[13]
    8: -> symbol: "TC:B4:SSISTS", comment: "Bottom SSI status", isConnectedTo: cover_elec.connectors.B4.pins[13]
  ) "slot3"


for [ slot, connector1, connector2, panel1, panel2 ] in [ ['slot4', 'T1', 'T2', 'Top 1', 'Top 2'],
                                                          ['slot5', 'T3', 'T4', 'Top 3', 'Top 4'],
                                                          ['slot6', 'B1', 'B2', 'Bottom 1', 'Bottom 2'],
                                                          ['slot7', 'B3', 'B4', 'Bottom 3', 'Bottom 4']]
  cover_elec.ADD IO_MODULE_INSTANCE(
    comment   : "SSI module for #{panel1} and #{panel2} encoders"
    type      : beckhoff.EL5002
    terminals :
      1: -> symbol: "TC:#{connector1}:SSID+", comment: "#{panel1} SSI encoder Data +" , isConnectedTo: cover_elec
      2: -> symbol: "TC:#{connector1}:SSIC+", comment: "#{panel1} SSI encoder Clock +", isConnectedTo: cover_elec
      3: -> symbol: "TC:#{connector2}:SSID+", comment: "#{panel2} SSI encoder Data +" , isConnectedTo: cover_elec
      4: -> symbol: "TC:#{connector2}:SSIC+", comment: "#{panel2} SSI encoder Clock +", isConnectedTo: cover_elec
      5: -> symbol: "TC:#{connector1}:SSID-", comment: "#{panel1} SSI encoder Data -" , isConnectedTo: cover_elec
      6: -> symbol: "TC:#{connector1}:SSIC-", comment: "#{panel1} SSI encoder Clock -", isConnectedTo: cover_elec
      7: -> symbol: "TC:#{connector2}:SSID-", comment: "#{panel2} SSI encoder Data -" , isConnectedTo: cover_elec
      8: -> symbol: "TC:#{connector2}:SSIC-", comment: "#{panel2} SSI encoder Clock -", isConnectedTo: cover_elec
  ) slot



```

How to build them?

- Example: model of an I/O module **instance**



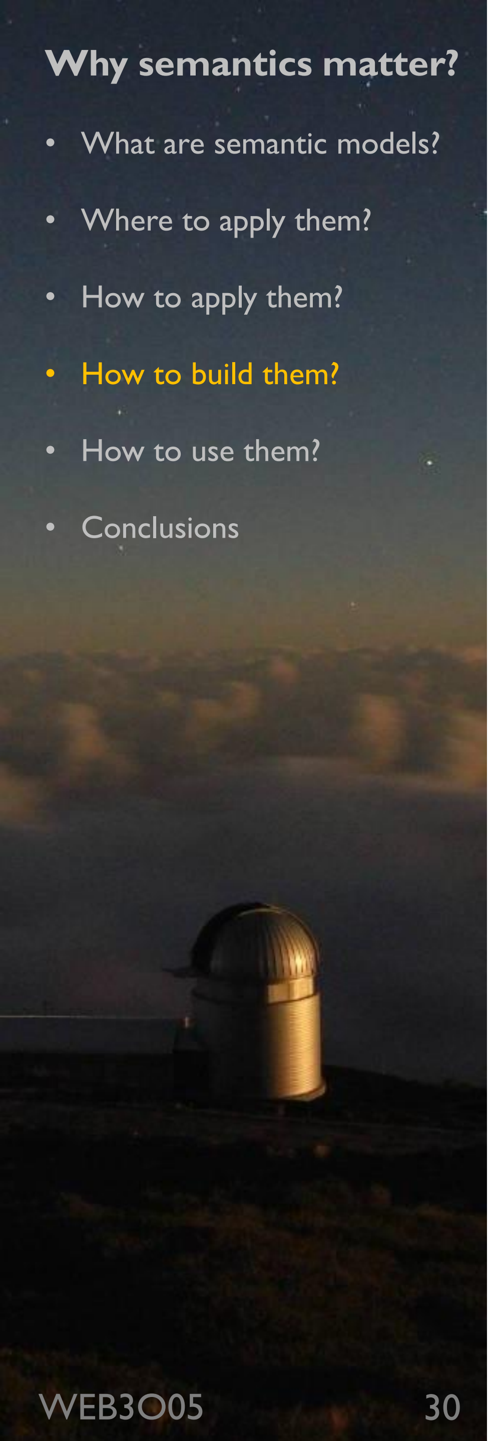
Models

- http://mercator.iac.es/onto/models/common/roles
- http://mercator.iac.es/onto/models/common/softwaremodels
- http://mercator.iac.es/onto/models/external/all
- http://mercator.iac.es/onto/models/external/beckhoff
- http://mercator.iac.es/onto/models/external/binder
- http://mercator.iac.es/onto/models/external/faulhaber
- http://mercator.iac.es/onto/models/external/harting
- http://mercator.iac.es/onto/models/external/itt
- http://mercator.iac.es/onto/models/external/maxon
- http://mercator.iac.es/onto/models/external/phoenix
- http://mercator.iac.es/onto/models/external/prehkeytec
- http://mercator.iac.es/onto/models/mtcs/common/all
- http://mercator.iac.es/onto/models/mtcs/common/hardware
- http://mercator.iac.es/onto/models/mtcs/common/software
- http://mercator.iac.es/onto/models/mtcs/common/statemachines
- http://mercator.iac.es/onto/models/mtcs/cover/electronics
- http://mercator.iac.es/onto/models/mtcs/cover/mechanics
- http://mercator.iac.es/onto/models/mtcs/cover/software
- http://mercator.iac.es/onto/models/mtcs/m1/electronics
- http://mercator.iac.es/onto/models/mtcs/m1/mechanics
- http://mercator.iac.es/onto/models/mtcs/m1/software
- http://mercator.iac.es/onto/models/mtcs/m3/electronics
- http://mercator.iac.es/onto/models/mtcs/m3/mechanics
- http://mercator.iac.es/onto/models/mtcs/m3/software
- http://mercator.iac.es/onto/models/mtcs/software
- http://mercator.iac.es/onto/models/mtcs/telemetry/electronics
- http://mercator.iac.es/onto/models/mtcs/telemetry/mechanics
- http://mercator.iac.es/onto/models/mtcs/telemetry/software
- http://mercator.iac.es/onto/models/mtcs/timing/electronics
- http://mercator.iac.es/onto/models/mtcs/timing/mechanics
- http://mercator.iac.es/onto/models/mtcs/timing/software
- http://mercator.iac.es/onto/models/test/test
- http://mercator.iac.es/onto/models/test/test_expressions

Model: http://mercator.iac.es/onto/models/mtcs/cover/electronics

Sources: [coffee]

```
cover_elec.ADD IO_MODULE_INSTANCE(
  type      : beckhoff.EL1088
  comment   : "Digital input terminal to read the status of the SSI encoders of all 8 cover panels"
  satisfies : cover_sys.panelDesign.requirements.absFeedbackStatus
  terminals :
    1: -> symbol: "TC:T1:SSISTS", comment: "Top 1 SSI status" , isConnectedTo: cover_elec.connectors.T1.pins[13]
    2: -> symbol: "TC:T2:SSISTS", comment: "Top 2 SSI status" , isConnectedTo: cover_elec.connectors.T2.pins[13]
    3: -> symbol: "TC:T3:SSISTS", comment: "Top 3 SSI status" , isConnectedTo: cover_elec.connectors.T3.pins[13]
    4: -> symbol: "TC:T4:SSISTS", comment: "Top 4 SSI status" , isConnectedTo: cover_elec.connectors.T4.pins[13]
    5: -> symbol: "TC:B1:SSISTS", comment: "Bottom SSI status", isConnectedTo: cover_elec.connectors.B1.pins[13]
    6: -> symbol: "TC:B2:SSISTS", comment: "Bottom SSI status", isConnectedTo: cover_elec.connectors.B2.pins[13]
    7: -> symbol: "TC:B3:SSISTS", comment: "Bottom SSI status", isConnectedTo: cover_elec.connectors.B3.pins[13]
    8: -> symbol: "TC:B4:SSISTS", comment: "Bottom SSI status", isConnectedTo: cover_elec.connectors.B4.pins[13]
  ) "slot3"


for [ slot, connector1, connector2, panel1, panel2 ] in [ ['slot4', 'T1', 'T2', 'Top 1', 'Top 2'],
                                                          ['slot5', 'T3', 'T4', 'Top 3', 'Top 4'],
                                                          ['slot6', 'B1', 'B2', 'Bottom 1', 'Bottom 2'],
                                                          ['slot7', 'B3', 'B4', 'Bottom 3', 'Bottom 4']]
  cover_elec.ADD IO_MODULE_INSTANCE(
    comment   : "SSI module for #{panel1} and #{panel2} encoders"
    type      : beckhoff.EL5002
    terminals :
      1: -> symbol: "TC:#{connector1}:SSID+", comment: "#{panel1} SSI encoder Data +" , isConnectedTo: cover_elec
      2: -> symbol: "TC:#{connector1}:SSIC+", comment: "#{panel1} SSI encoder Clock +", isConnectedTo: cover_elec
      3: -> symbol: "TC:#{connector2}:SSID+", comment: "#{panel2} SSI encoder Data +" , isConnectedTo: cover_elec
      4: -> symbol: "TC:#{connector2}:SSIC+", comment: "#{panel2} SSI encoder Clock +", isConnectedTo: cover_elec
      5: -> symbol: "TC:#{connector1}:SSID-", comment: "#{panel1} SSI encoder Data -" , isConnectedTo: cover_elec
      6: -> symbol: "TC:#{connector1}:SSIC-", comment: "#{panel1} SSI encoder Clock -", isConnectedTo: cover_elec
      7: -> symbol: "TC:#{connector2}:SSID-", comment: "#{panel2} SSI encoder Data -" , isConnectedTo: cover_elec
      8: -> symbol: "TC:#{connector2}:SSIC-", comment: "#{panel2} SSI encoder Clock -", isConnectedTo: cover_elec
  ) slot
```

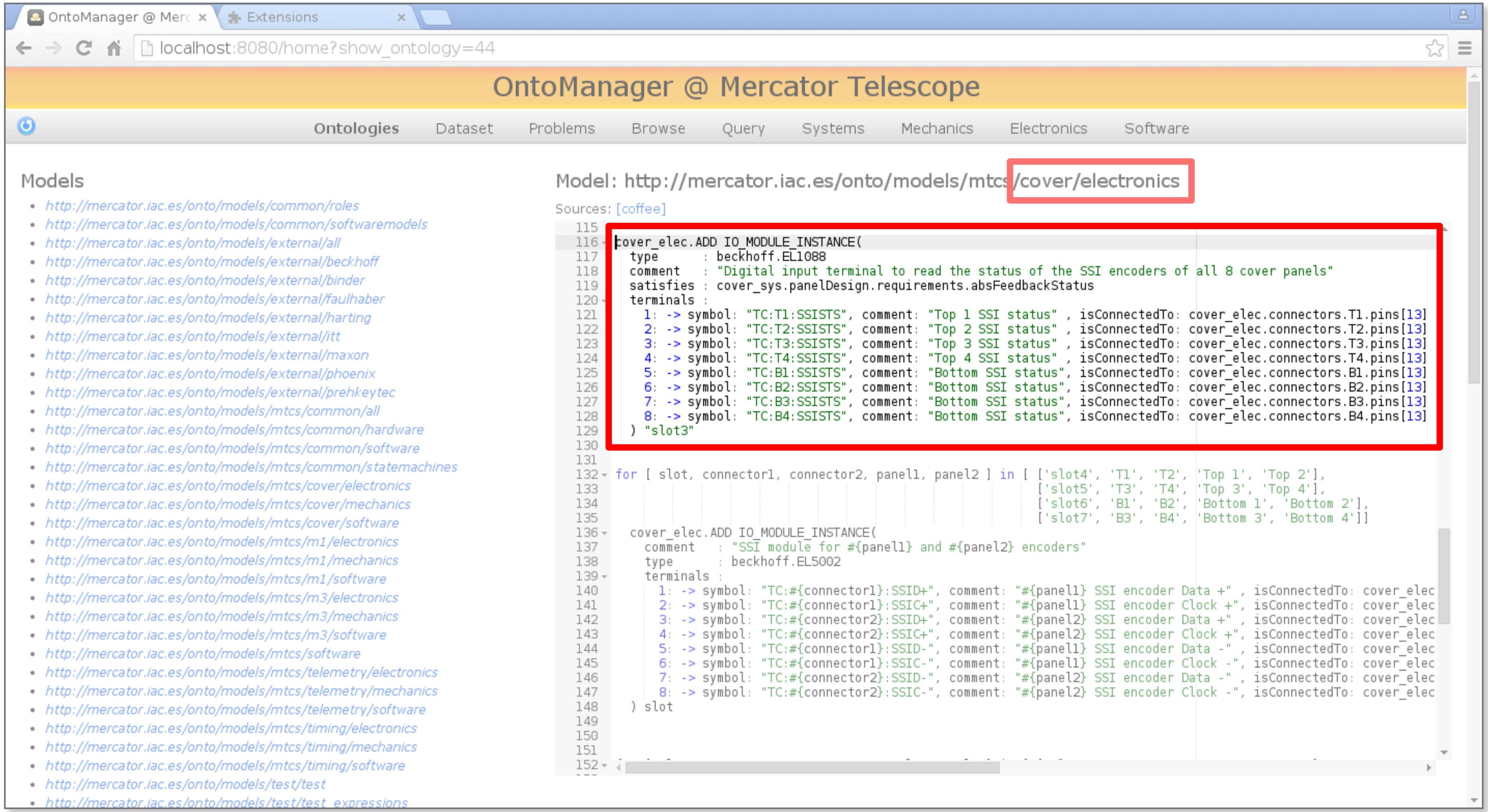


Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

How to build them?

- Example: model of an I/O module **instance**



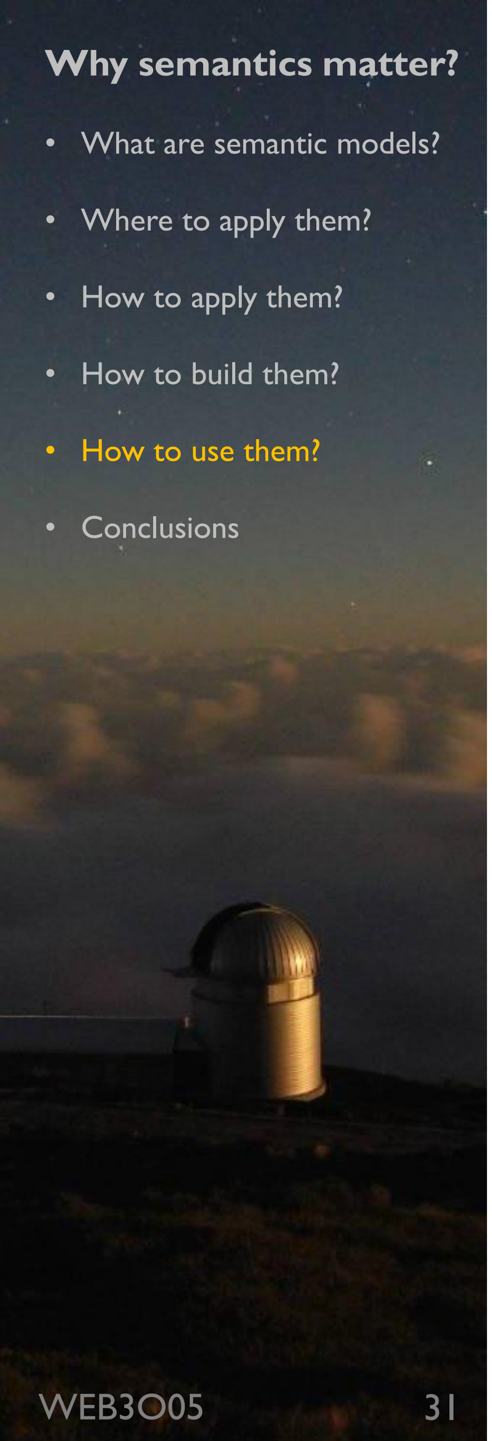
Models

- http://mercator.iac.es/onto/models/common/roles
- http://mercator.iac.es/onto/models/common/softwaremodels
- http://mercator.iac.es/onto/models/external/all
- http://mercator.iac.es/onto/models/external/beckhoff
- http://mercator.iac.es/onto/models/external/binder
- http://mercator.iac.es/onto/models/external/faulhaber
- http://mercator.iac.es/onto/models/external/harting
- http://mercator.iac.es/onto/models/external/itt
- http://mercator.iac.es/onto/models/external/maxon
- http://mercator.iac.es/onto/models/external/phoenix
- http://mercator.iac.es/onto/models/external/prehkeytec
- http://mercator.iac.es/onto/models/mtcs/common/all
- http://mercator.iac.es/onto/models/mtcs/common/hardware
- http://mercator.iac.es/onto/models/mtcs/common/software
- http://mercator.iac.es/onto/models/mtcs/common/statemachines
- http://mercator.iac.es/onto/models/mtcs/cover/electronics
- http://mercator.iac.es/onto/models/mtcs/cover/mechanics
- http://mercator.iac.es/onto/models/mtcs/cover/software
- http://mercator.iac.es/onto/models/mtcs/m1/electronics
- http://mercator.iac.es/onto/models/mtcs/m1/mechanics
- http://mercator.iac.es/onto/models/mtcs/m1/software
- http://mercator.iac.es/onto/models/mtcs/m3/electronics
- http://mercator.iac.es/onto/models/mtcs/m3/mechanics
- http://mercator.iac.es/onto/models/mtcs/m3/software
- http://mercator.iac.es/onto/models/mtcs/software
- http://mercator.iac.es/onto/models/mtcs/telemetry/electronics
- http://mercator.iac.es/onto/models/mtcs/telemetry/mechanics
- http://mercator.iac.es/onto/models/mtcs/telemetry/software
- http://mercator.iac.es/onto/models/mtcs/timing/electronics
- http://mercator.iac.es/onto/models/mtcs/timing/mechanics
- http://mercator.iac.es/onto/models/mtcs/timing/software
- http://mercator.iac.es/onto/models/test/test
- http://mercator.iac.es/onto/models/test/test_expressions

Model: http://mercator.iac.es/onto/models/mtcs/cover/electronics

Sources: [coffee]

```
cover_elec.ADD IO_MODULE_INSTANCE(
  type      : beckhoff.EL1088
  comment   : "Digital input terminal to read the status of the SSI encoders of all 8 cover panels"
  satisfies : cover_sys.panelDesign.requirements.absFeedbackStatus
  terminals :
    1: -> symbol: "TC:T1:SSISTS", comment: "Top 1 SSI status" , isConnectedTo: cover_elec.connectors.T1.pins[13]
    2: -> symbol: "TC:T2:SSISTS", comment: "Top 2 SSI status" , isConnectedTo: cover_elec.connectors.T2.pins[13]
    3: -> symbol: "TC:T3:SSISTS", comment: "Top 3 SSI status" , isConnectedTo: cover_elec.connectors.T3.pins[13]
    4: -> symbol: "TC:T4:SSISTS", comment: "Top 4 SSI status" , isConnectedTo: cover_elec.connectors.T4.pins[13]
    5: -> symbol: "TC:B1:SSISTS", comment: "Bottom SSI status", isConnectedTo: cover_elec.connectors.B1.pins[13]
    6: -> symbol: "TC:B2:SSISTS", comment: "Bottom SSI status", isConnectedTo: cover_elec.connectors.B2.pins[13]
    7: -> symbol: "TC:B3:SSISTS", comment: "Bottom SSI status", isConnectedTo: cover_elec.connectors.B3.pins[13]
    8: -> symbol: "TC:B4:SSISTS", comment: "Bottom SSI status", isConnectedTo: cover_elec.connectors.B4.pins[13]
  ) "slot3"


for [ slot, connector1, connector2, panel1, panel2 ] in [ ['slot4', 'T1', 'T2', 'Top 1', 'Top 2'],
                                                          ['slot5', 'T3', 'T4', 'Top 3', 'Top 4'],
                                                          ['slot6', 'B1', 'B2', 'Bottom 1', 'Bottom 2'],
                                                          ['slot7', 'B3', 'B4', 'Bottom 3', 'Bottom 4']]
  cover_elec.ADD IO_MODULE_INSTANCE(
    comment   : "SSI module for #{panel1} and #{panel2} encoders"
    type      : beckhoff.EL5002
    terminals :
      1: -> symbol: "TC:#{connector1}:SSID+", comment: "#{panel1} SSI encoder Data +" , isConnectedTo: cover_elec
      2: -> symbol: "TC:#{connector1}:SSIC+", comment: "#{panel1} SSI encoder Clock +", isConnectedTo: cover_elec
      3: -> symbol: "TC:#{connector2}:SSID+", comment: "#{panel2} SSI encoder Data +" , isConnectedTo: cover_elec
      4: -> symbol: "TC:#{connector2}:SSIC+", comment: "#{panel2} SSI encoder Clock +", isConnectedTo: cover_elec
      5: -> symbol: "TC:#{connector1}:SSID-", comment: "#{panel1} SSI encoder Data -" , isConnectedTo: cover_elec
      6: -> symbol: "TC:#{connector1}:SSIC-", comment: "#{panel1} SSI encoder Clock -", isConnectedTo: cover_elec
      7: -> symbol: "TC:#{connector2}:SSID-", comment: "#{panel2} SSI encoder Data -" , isConnectedTo: cover_elec
      8: -> symbol: "TC:#{connector2}:SSIC-", comment: "#{panel2} SSI encoder Clock -", isConnectedTo: cover_elec
  ) slot
```



Why semantics matter?

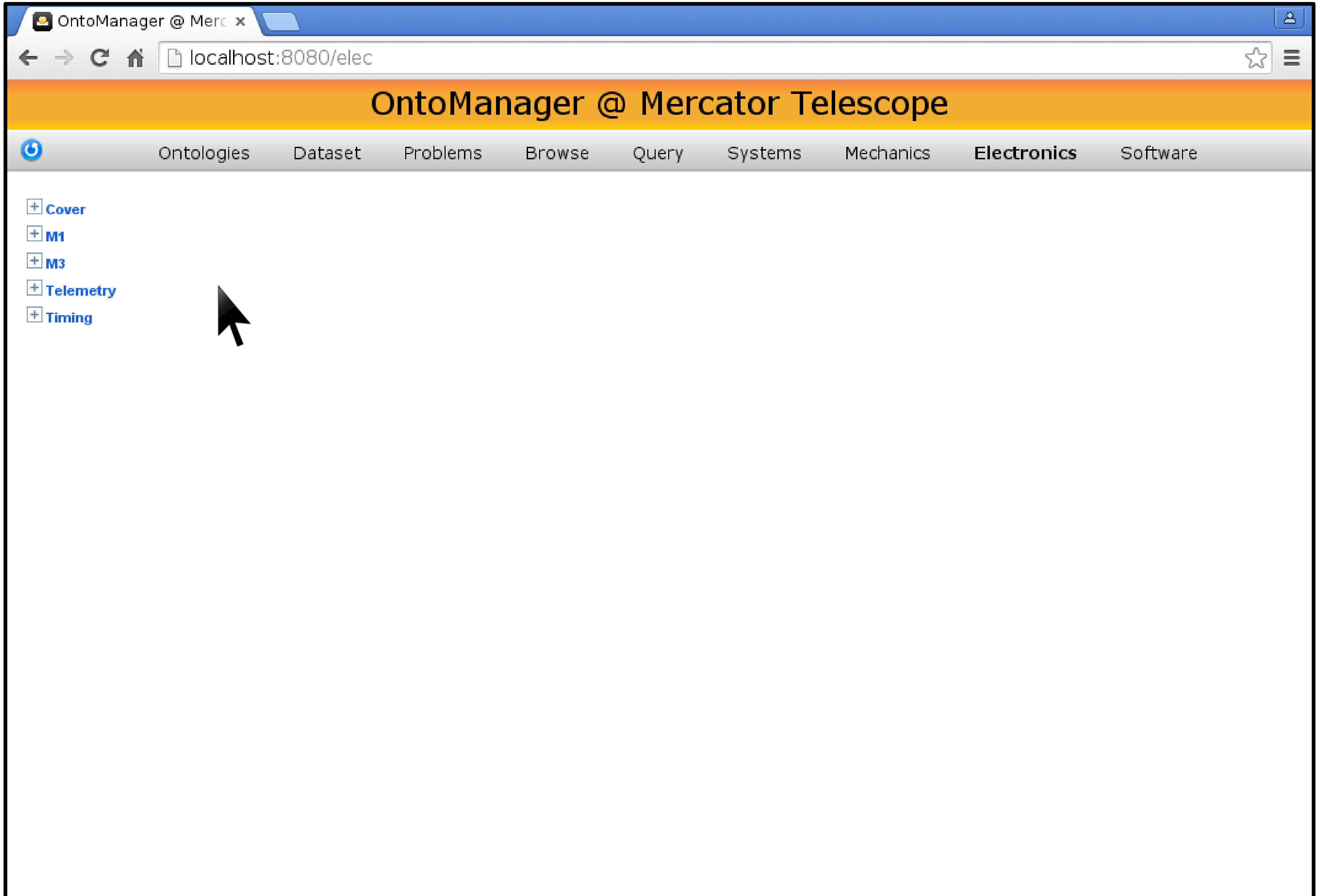
- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

How to use them?

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Electrical design


OntoManager @ Merc

localhost:8080/elec

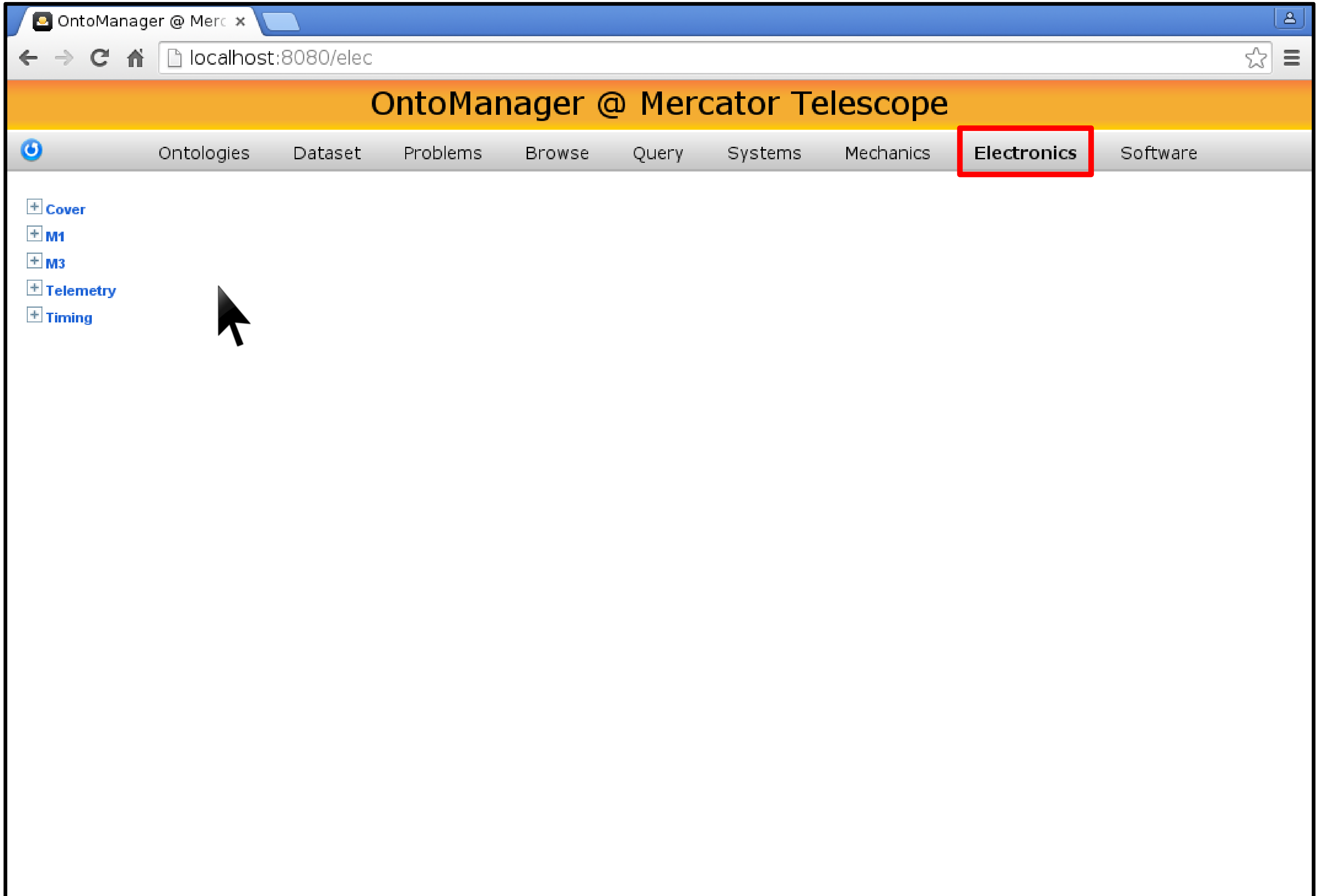
OntoManager @ Mercator Telescope

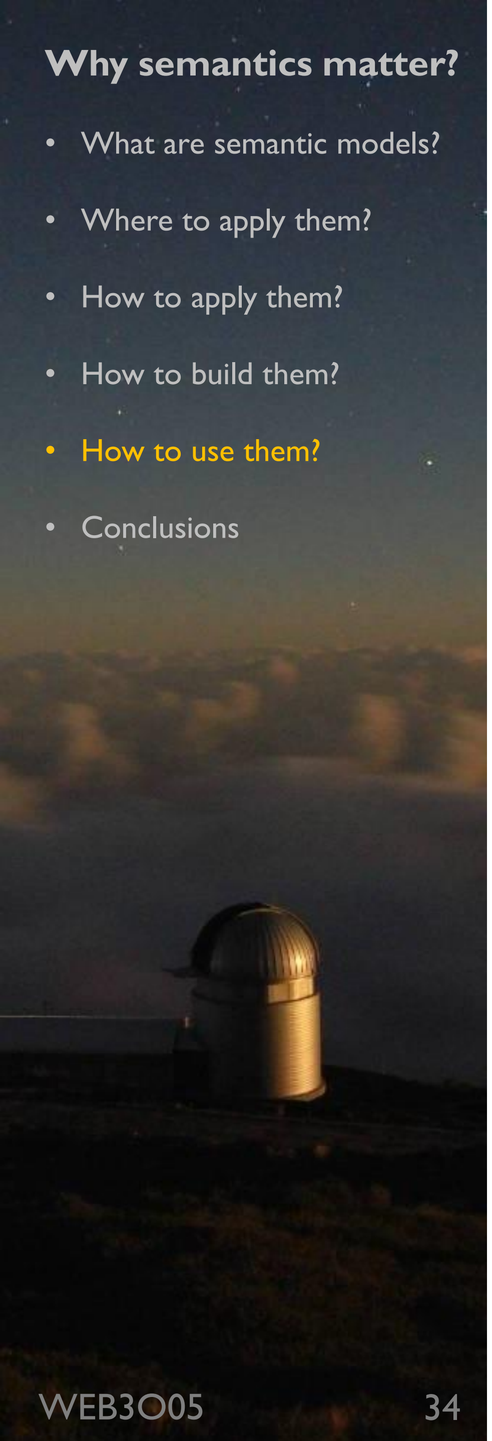
⊞ Cover
⊞ M1
⊞ M3
⊞ Telemetry
⊞ Timing

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Electrical design

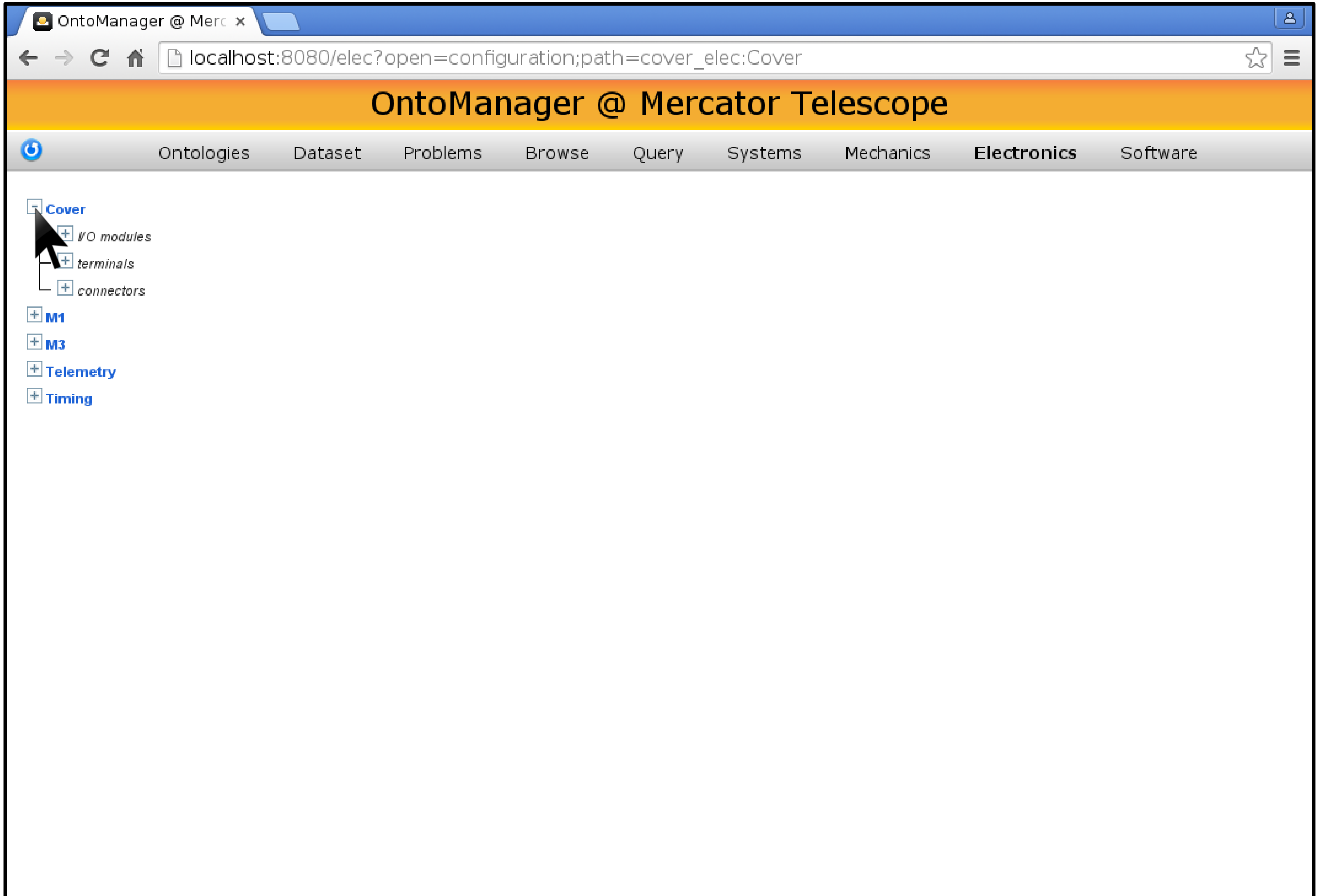




Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Electrical design



OntoManager @ Merc ×
localhost:8080/elec?open=configuration;path=cover_elec:Cover

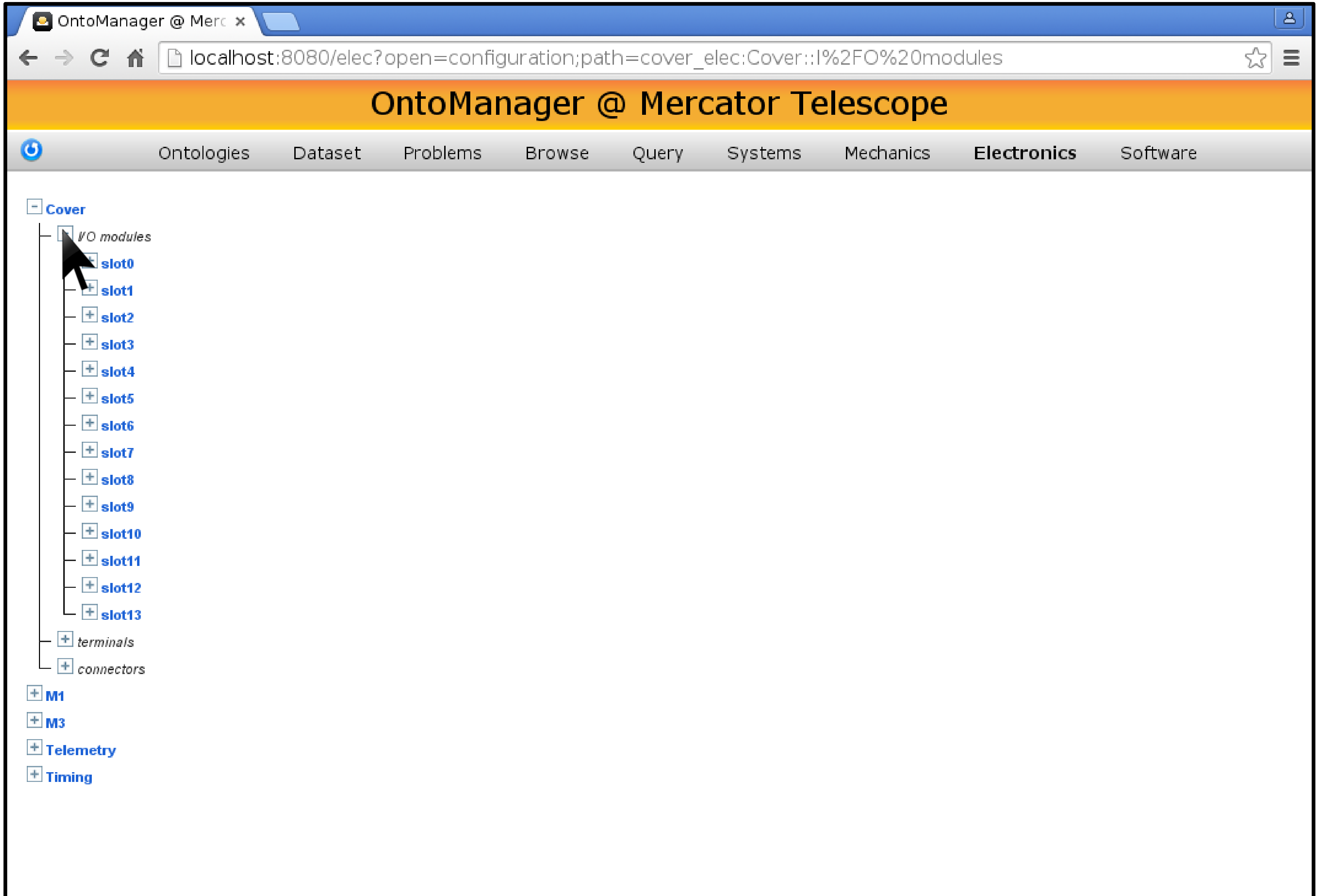
OntoManager @ Mercator Telescope

- Cover
  - I/O modules
  - terminals
  - connectors
- M1
- M3
- Telemetry
- Timing

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

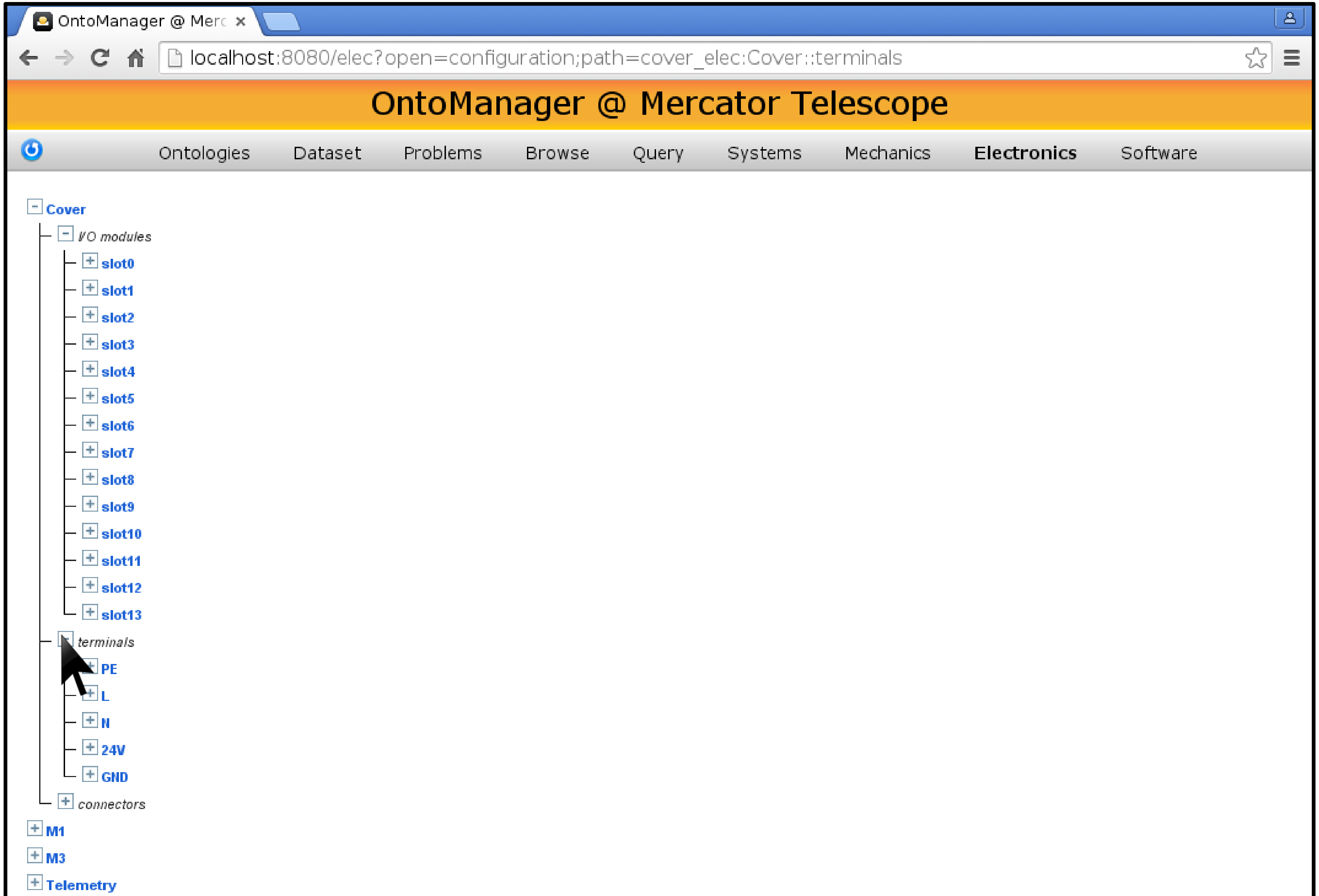
Electrical design



Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Electrical design



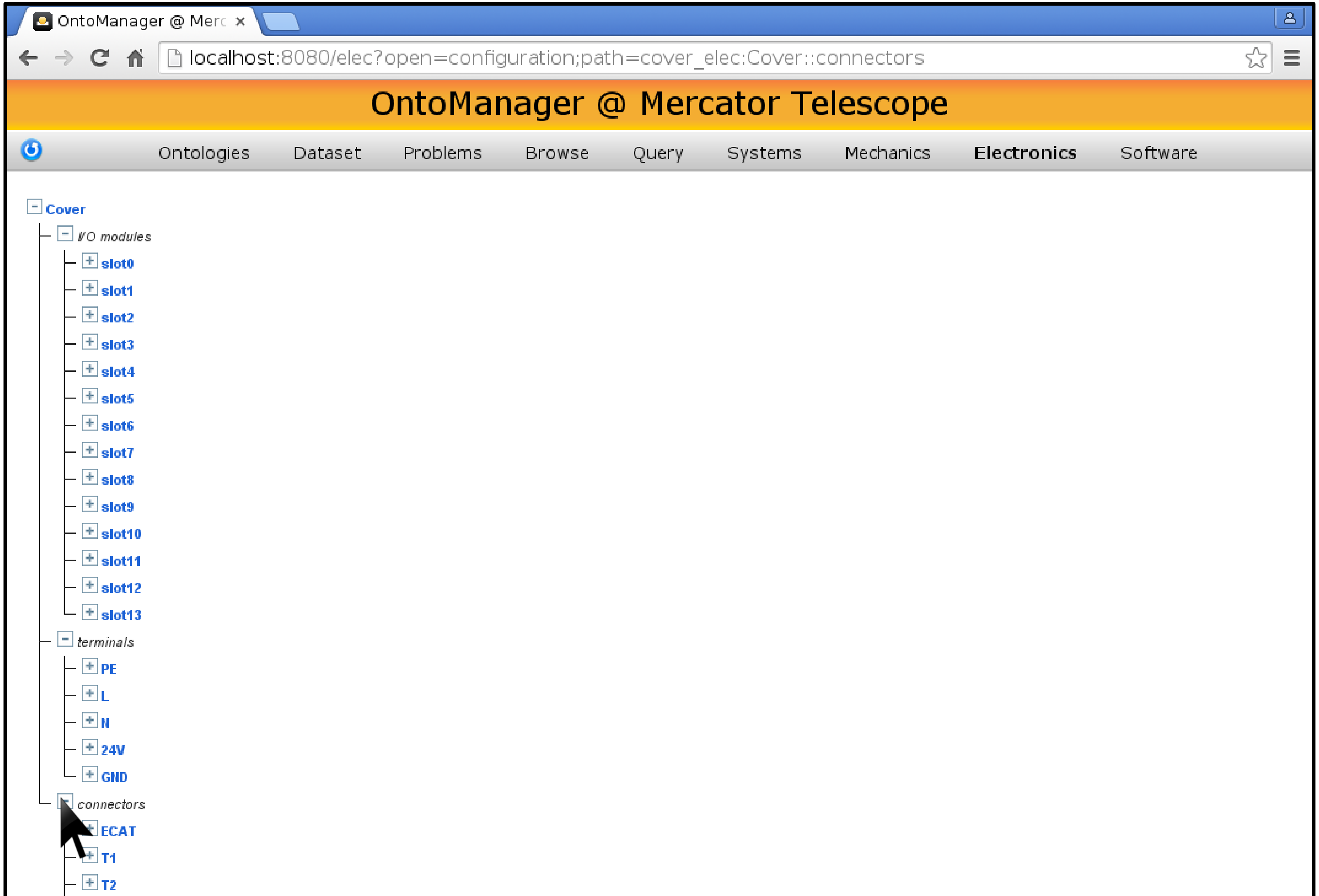
OntoManager @ Merc
localhost:8080/elec?open=configuration;path=cover_elec:Cover::terminals
OntoManager @ Mercator Telescope
Ontologies
Dataset
Problems
Browse
Query
Systems
Mechanics
Electronics
Software

- Cover
  - *I/O modules*
    - slot0
    - slot1
    - slot2
    - slot3
    - slot4
    - slot5
    - slot6
    - slot7
    - slot8
    - slot9
    - slot10
    - slot11
    - slot12
    - slot13
  - *terminals*
    - PE
    - L
    - N
    - 24V
    - GND
  - *connectors*
- M1
- M3
- Telemetry

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Electrical design



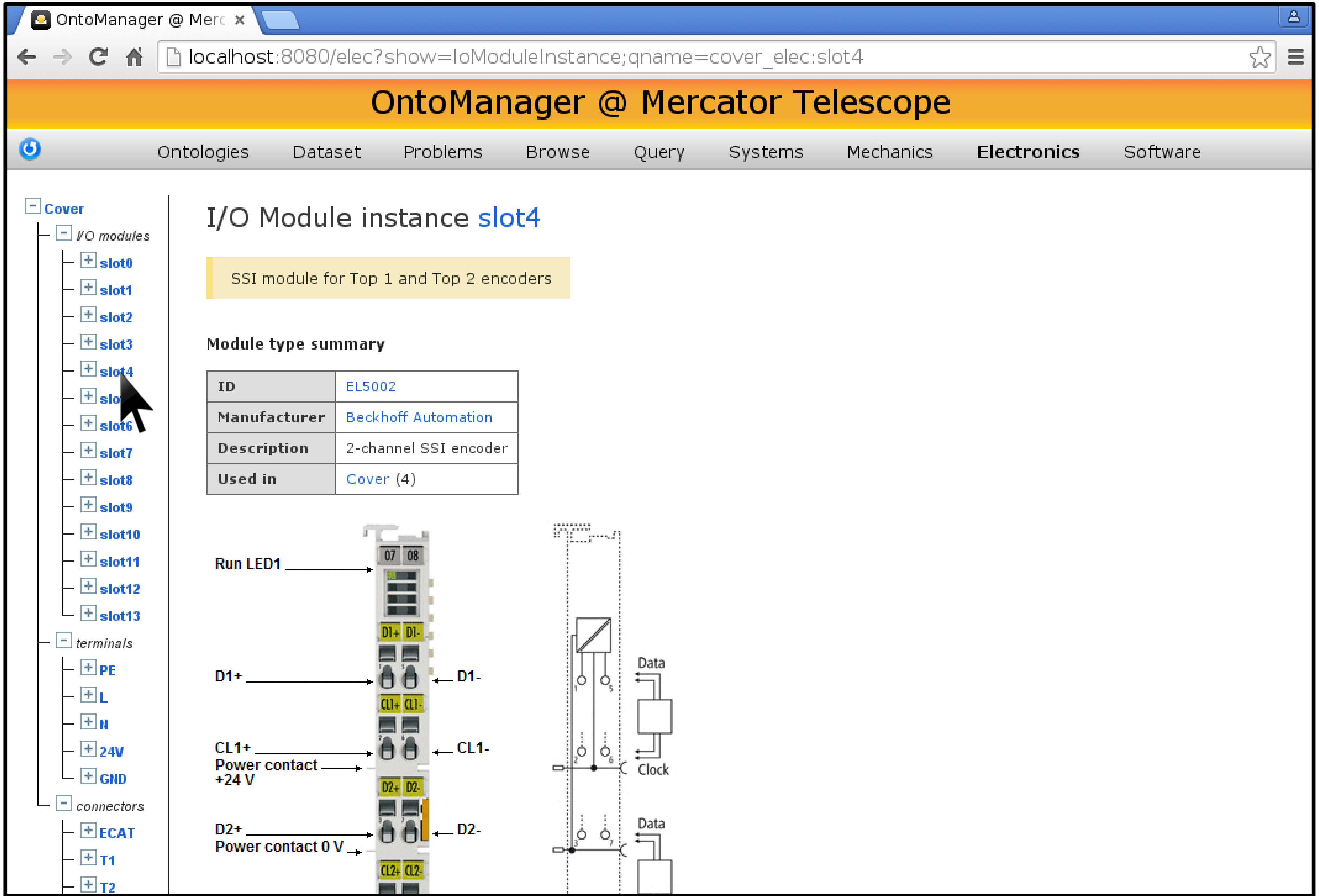
OntoManager @ Merc
localhost:8080/elec?open=configuration;path=cover_elec:Cover::connectors
OntoManager @ Mercator Telescope
Ontologies
Dataset
Problems
Browse
Query
Systems
Mechanics
Electronics
Software

- Cover
  - I/O modules
    - slot0
    - slot1
    - slot2
    - slot3
    - slot4
    - slot5
    - slot6
    - slot7
    - slot8
    - slot9
    - slot10
    - slot11
    - slot12
    - slot13
  - terminals
    - PE
    - L
    - N
    - 24V
    - GND
  - connectors
    - ECAT
    - T1
    - T2

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Electrical design



OntoManager @ Merc ×

localhost:8080/elec?show=IoModuleInstance;qname=cover_elec:slot4

OntoManager @ Mercator Telescope

Ontologies Dataset Problems Browse Query Systems Mechanics **Electronics** Software

- Cover
  - *I/O modules*
    - slot0
    - slot1
    - slot2
    - slot3
    - slot4
    - slo
    - slot6
    - slot7
    - slot8
    - slot9
    - slot10
    - slot11
    - slot12
    - slot13
  - *terminals*
    - PE
    - L
    - N
    - 24V
    - GND
  - *connectors*
    - ECAT
    - T1
    - T2

I/O Module instance slot4

SSI module for Top 1 and Top 2 encoders

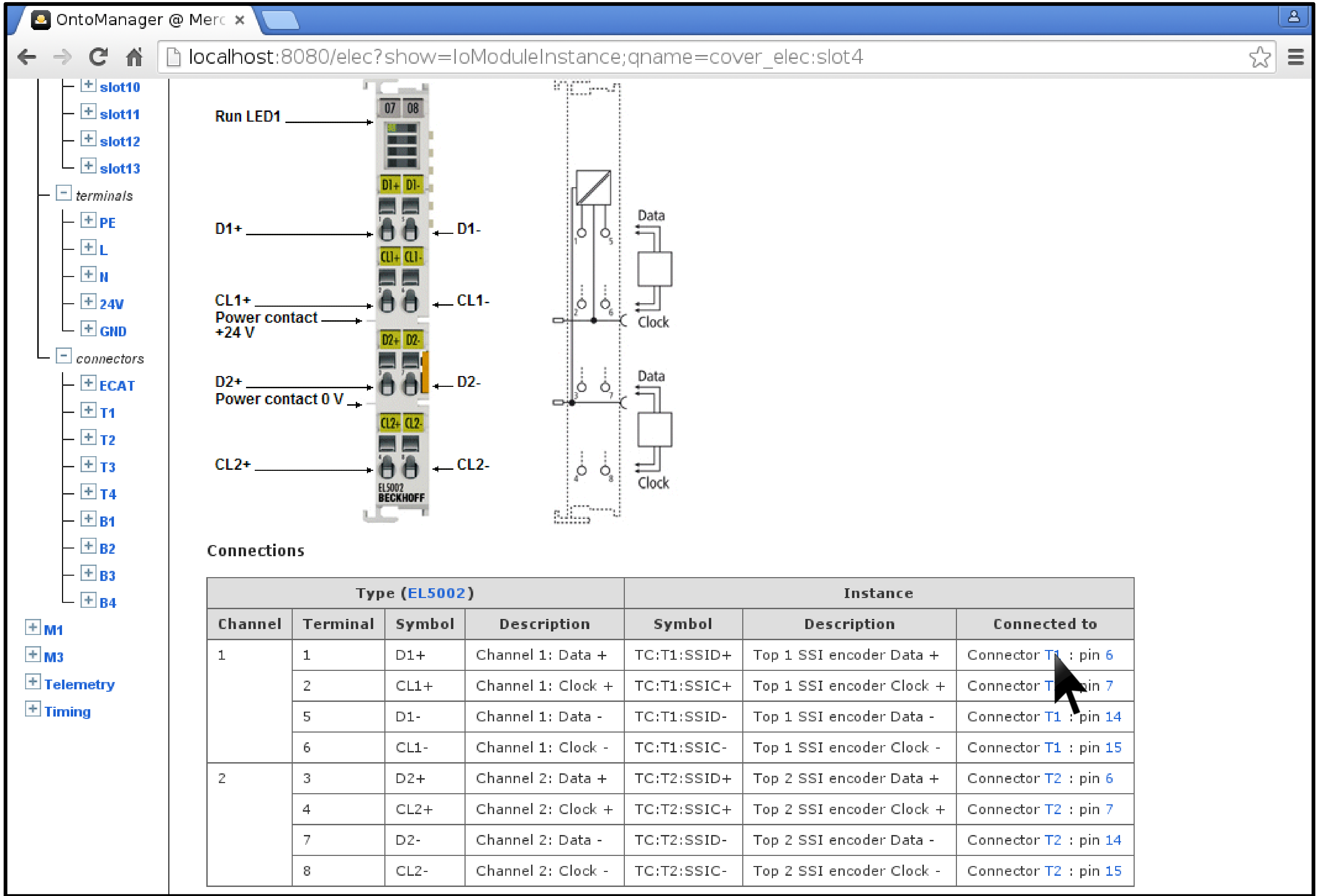
Module type summary

ID	EL5002
Manufacturer	Beckhoff Automation
Description	2-channel SSI encoder
Used in	Cover (4)

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Electrical design



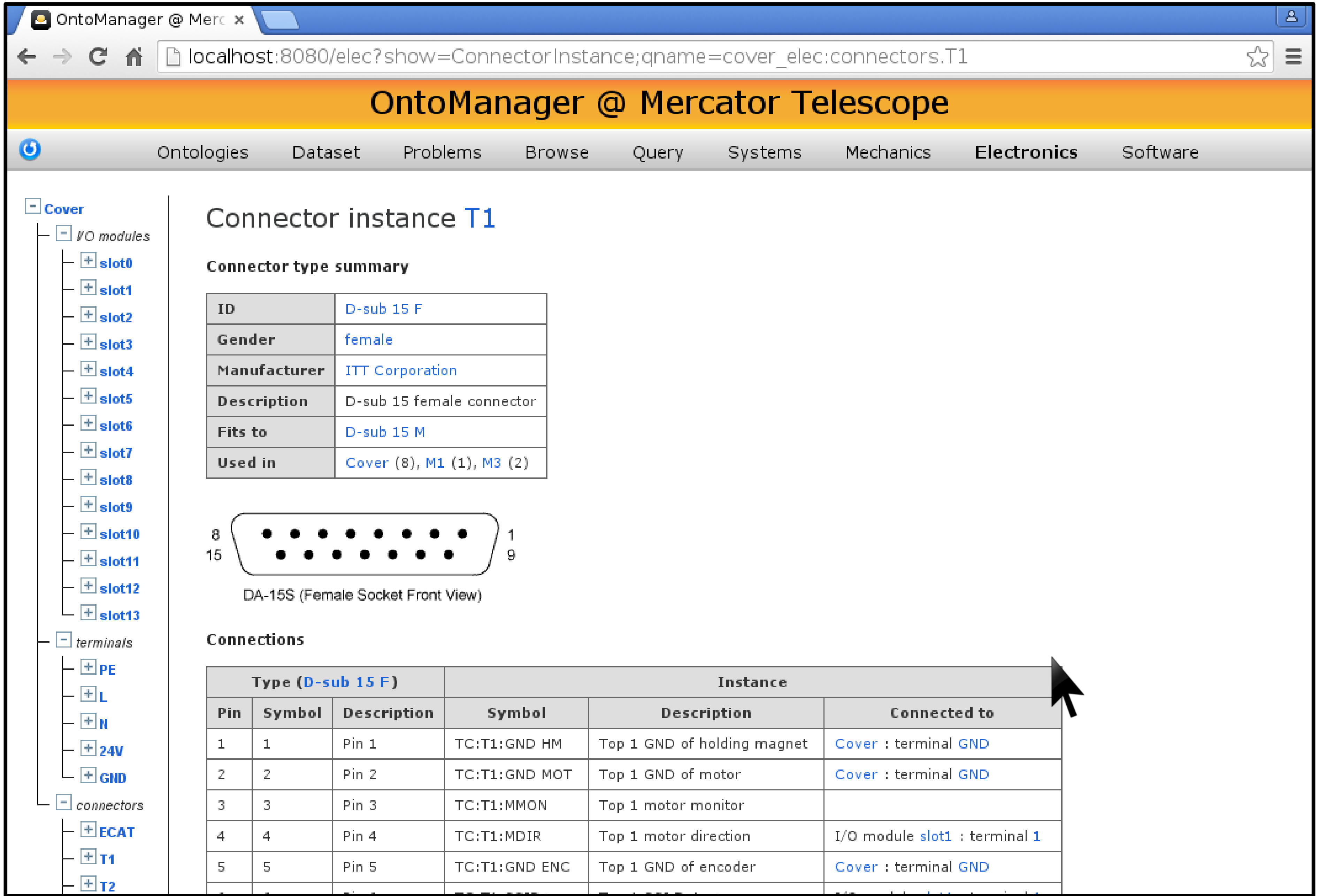
Connections

Type (EL5002)				Instance		
Channel	Terminal	Symbol	Description	Symbol	Description	Connected to
1	1	D1+	Channel 1: Data +	TC:T1:SSID+	Top 1 SSI encoder Data +	Connector T1 : pin 6
	2	CL1+	Channel 1: Clock +	TC:T1:SSIC+	Top 1 SSI encoder Clock +	Connector T1 : pin 7
	5	D1-	Channel 1: Data -	TC:T1:SSID-	Top 1 SSI encoder Data -	Connector T1 : pin 14
	6	CL1-	Channel 1: Clock -	TC:T1:SSIC-	Top 1 SSI encoder Clock -	Connector T1 : pin 15
2	3	D2+	Channel 2: Data +	TC:T2:SSID+	Top 2 SSI encoder Data +	Connector T2 : pin 6
	4	CL2+	Channel 2: Clock +	TC:T2:SSIC+	Top 2 SSI encoder Clock +	Connector T2 : pin 7
	7	D2-	Channel 2: Data -	TC:T2:SSID-	Top 2 SSI encoder Data -	Connector T2 : pin 14
	8	CL2-	Channel 2: Clock -	TC:T2:SSIC-	Top 2 SSI encoder Clock -	Connector T2 : pin 15

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Electrical design



OntoManager @ Merc
localhost:8080/elec?show=ConnectorInstance;qname=cover_elec:connectors.T1

OntoManager @ Mercator Telescope

Connector instance T1

Connector type summary

ID	D-sub 15 F
Gender	female
Manufacturer	ITT Corporation
Description	D-sub 15 female connector
Fits to	D-sub 15 M
Used in	Cover (8), M1 (1), M3 (2)

DA-15S (Female Socket Front View)

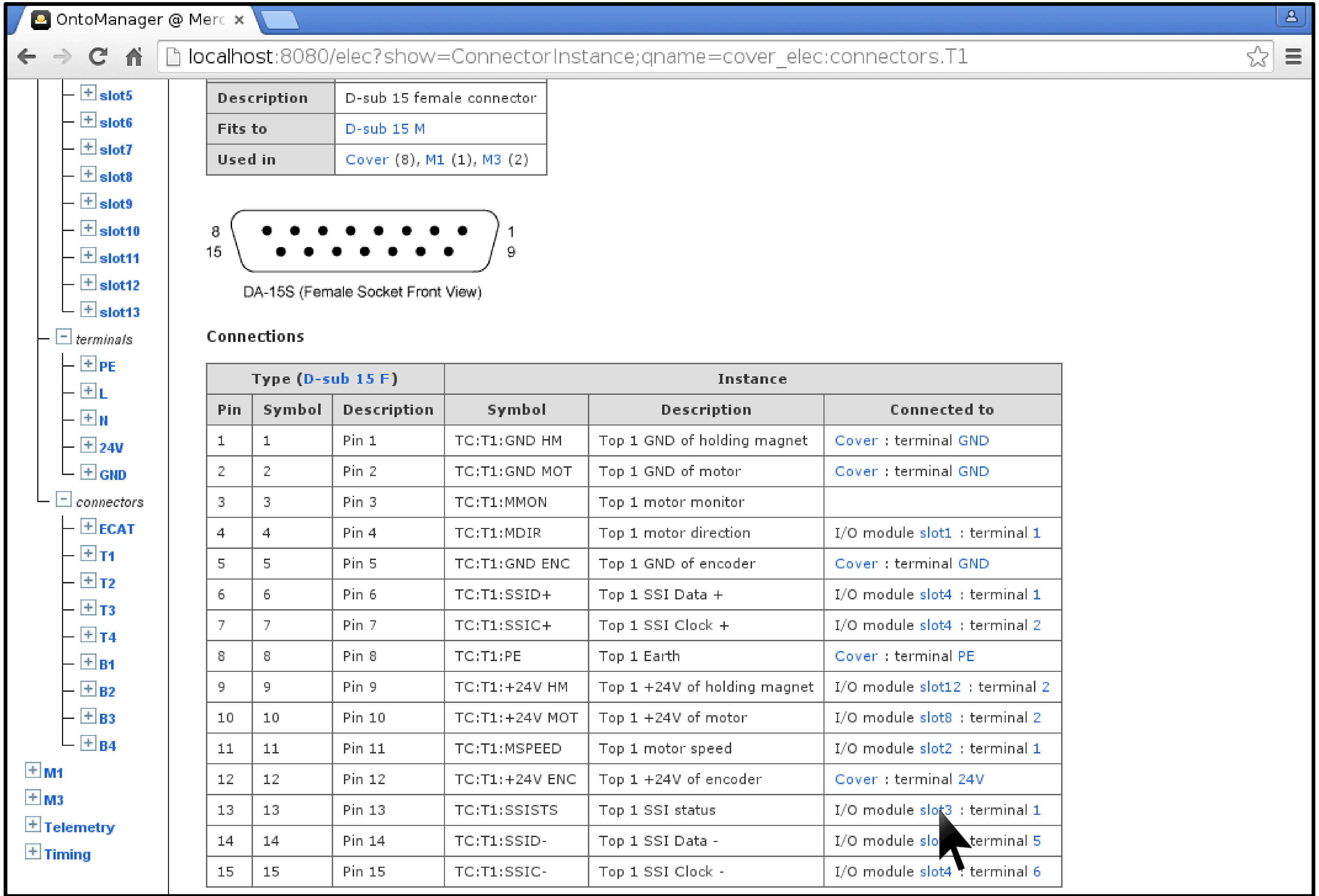
Connections

Type (D-sub 15 F)			Instance		
Pin	**Symbol**	**Description**	**Symbol**	**Description**	**Connected to**
1	1	Pin 1	TC:T1:GND HM	Top 1 GND of holding magnet	Cover : terminal GND
2	2	Pin 2	TC:T1:GND MOT	Top 1 GND of motor	Cover : terminal GND
3	3	Pin 3	TC:T1:MMON	Top 1 motor monitor	
4	4	Pin 4	TC:T1:MDIR	Top 1 motor direction	I/O module slot1 : terminal 1
5	5	Pin 5	TC:T1:GND ENC	Top 1 GND of encoder	Cover : terminal GND

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Electrical design



Description	D-sub 15 female connector
Fits to	D-sub 15 M
Used in	Cover (8), M1 (1), M3 (2)

DA-15S (Female Socket Front View)

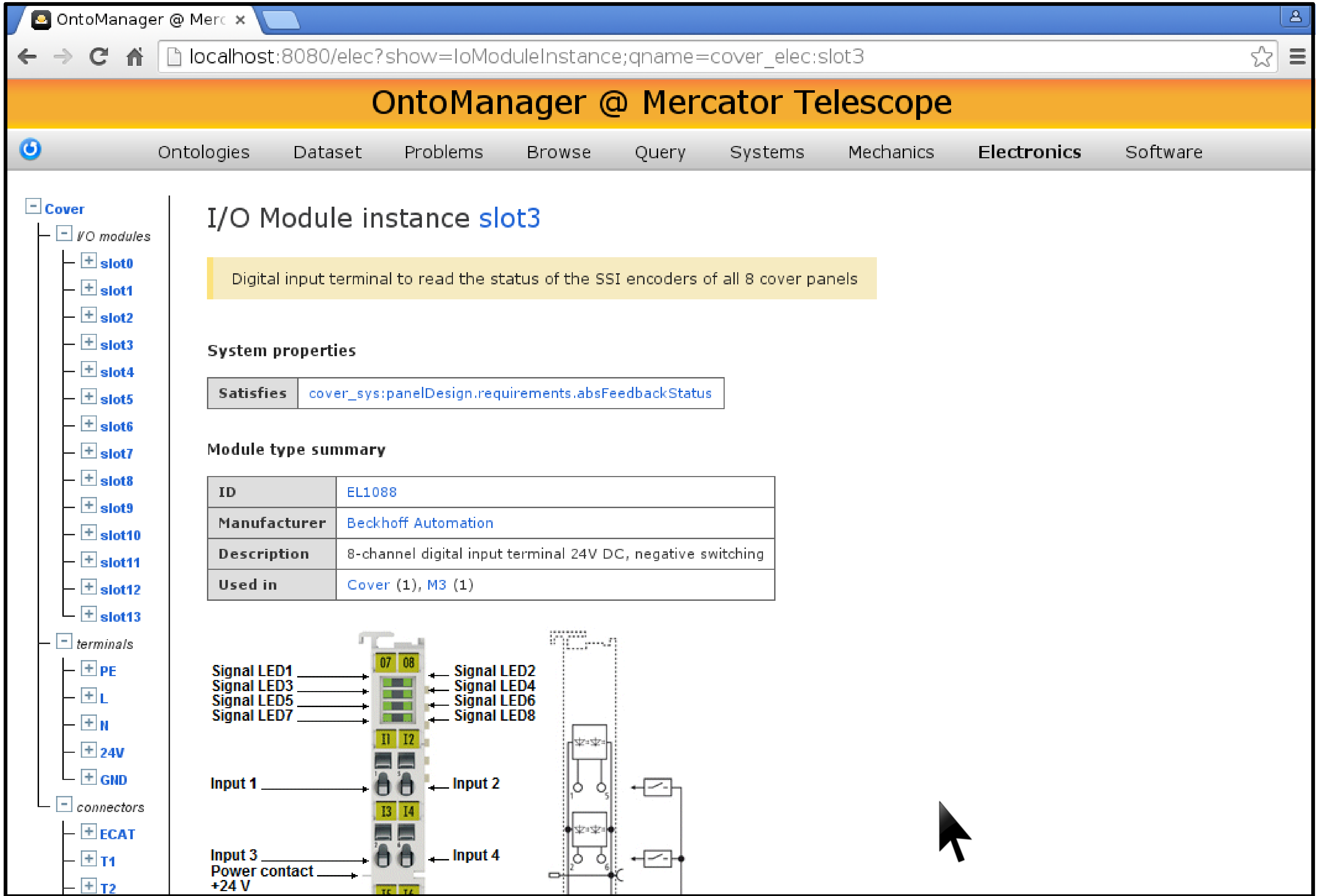
Connections

Type (D-sub 15 F)			Instance		
Pin	Symbol	Description	Symbol	Description	Connected to
1	1	Pin 1	TC:T1:GND HM	Top 1 GND of holding magnet	Cover : terminal GND
2	2	Pin 2	TC:T1:GND MOT	Top 1 GND of motor	Cover : terminal GND
3	3	Pin 3	TC:T1:MMON	Top 1 motor monitor	
4	4	Pin 4	TC:T1:MDIR	Top 1 motor direction	I/O module slot1 : terminal 1
5	5	Pin 5	TC:T1:GND ENC	Top 1 GND of encoder	Cover : terminal GND
6	6	Pin 6	TC:T1:SSID+	Top 1 SSI Data +	I/O module slot4 : terminal 1
7	7	Pin 7	TC:T1:SSIC+	Top 1 SSI Clock +	I/O module slot4 : terminal 2
8	8	Pin 8	TC:T1:PE	Top 1 Earth	Cover : terminal PE
9	9	Pin 9	TC:T1:+24V HM	Top 1 +24V of holding magnet	I/O module slot12 : terminal 2
10	10	Pin 10	TC:T1:+24V MOT	Top 1 +24V of motor	I/O module slot8 : terminal 2
11	11	Pin 11	TC:T1:MSPEED	Top 1 motor speed	I/O module slot2 : terminal 1
12	12	Pin 12	TC:T1:+24V ENC	Top 1 +24V of encoder	Cover : terminal 24V
13	13	Pin 13	TC:T1:SSISTS	Top 1 SSI status	I/O module slot3 : terminal 1
14	14	Pin 14	TC:T1:SSID-	Top 1 SSI Data -	I/O module slo terminal 5
15	15	Pin 15	TC:T1:SSIC-	Top 1 SSI Clock -	I/O module slot4 : terminal 6

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Electrical design



OntoManager @ Merc
localhost:8080/elec?show=IoModuleInstance;qname=cover_elec:slot3

OntoManager @ Mercator Telescope

I/O Module instance slot3

Digital input terminal to read the status of the SSI encoders of all 8 cover panels

System properties

Satisfies	cover_sys:panelDesign.requirements.absFeedbackStatus

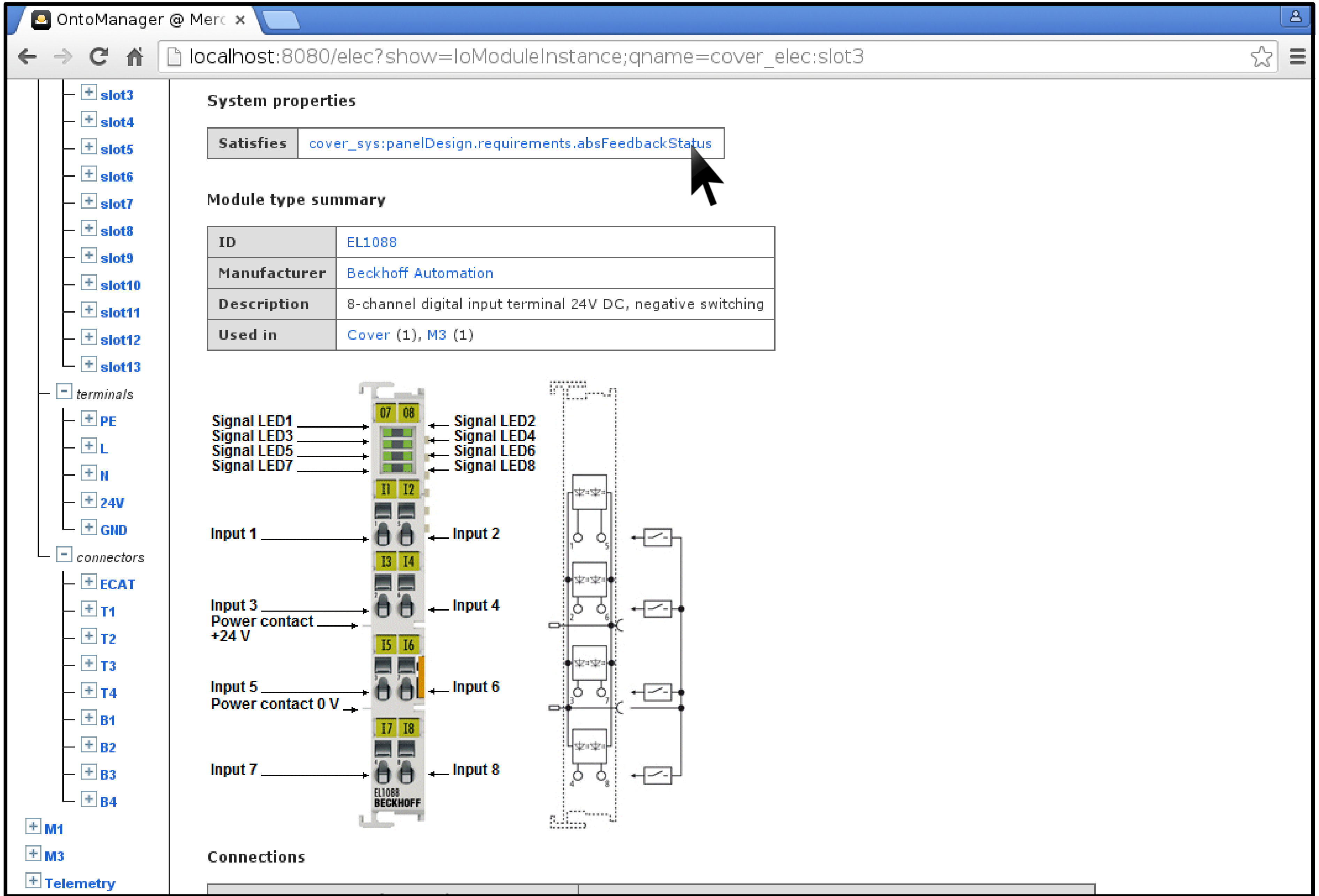
Module type summary

ID	EL1088
Manufacturer	Beckhoff Automation
Description	8-channel digital input terminal 24V DC, negative switching
Used in	Cover (1), M3 (1)

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Electrical design



System properties

Satisfies	cover_sys:panelDesign.requirements.absFeedbackStatus

Module type summary

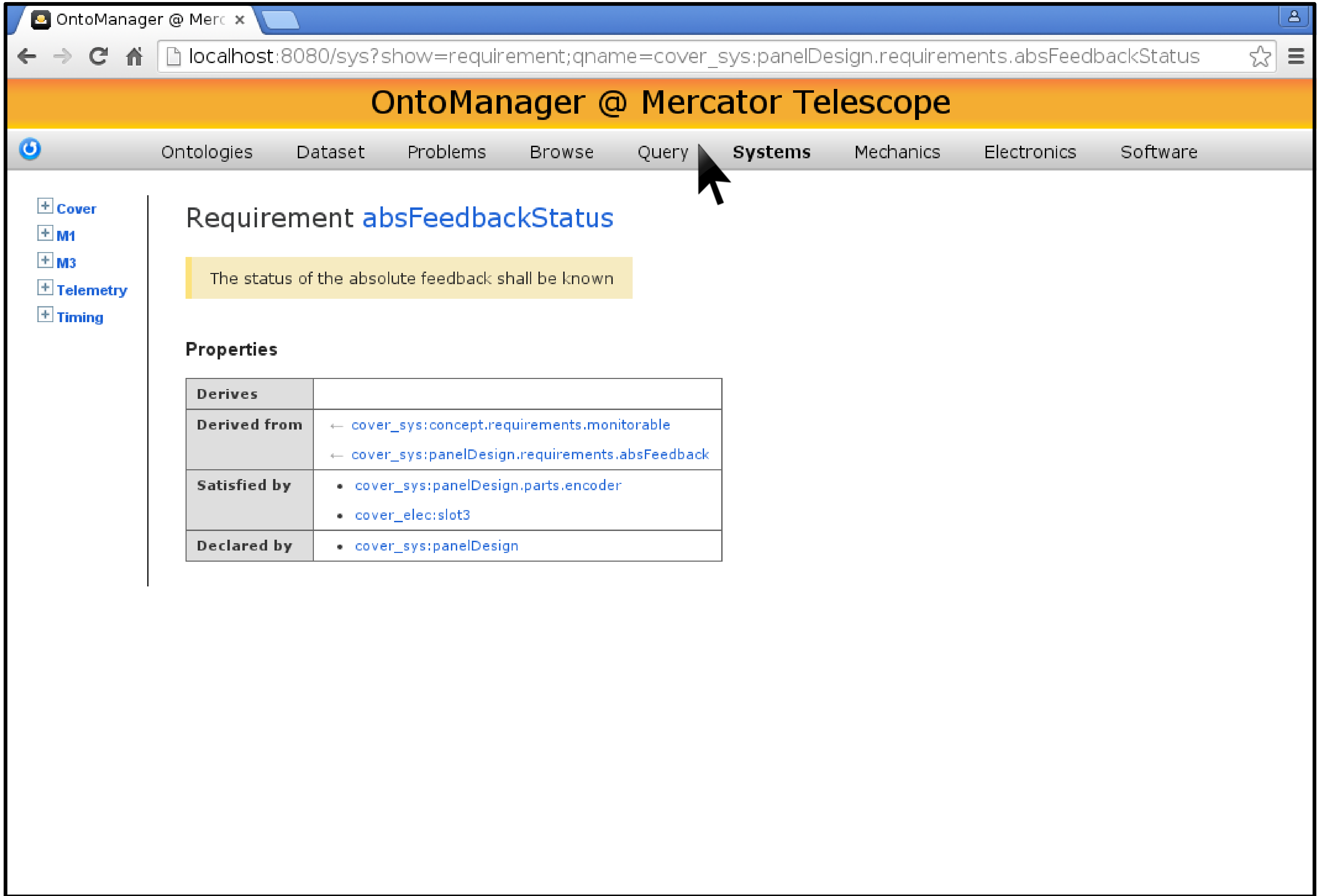
ID	EL1088
Manufacturer	Beckhoff Automation
Description	8-channel digital input terminal 24V DC, negative switching
Used in	Cover (1), M3 (1)

Connections

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Systems design


OntoManager @ Merc ×
localhost:8080/sys?show=requirement;qname=cover_sys:panelDesign.requirements.absFeedbackStatus
OntoManager @ Mercator Telescope
Ontologies
Dataset
Problems
Browse
Query
Systems
Mechanics
Electronics
Software
Cover
M1
M3
Telemetry
Timing

Requirement absFeedbackStatus

The status of the absolute feedback shall be known

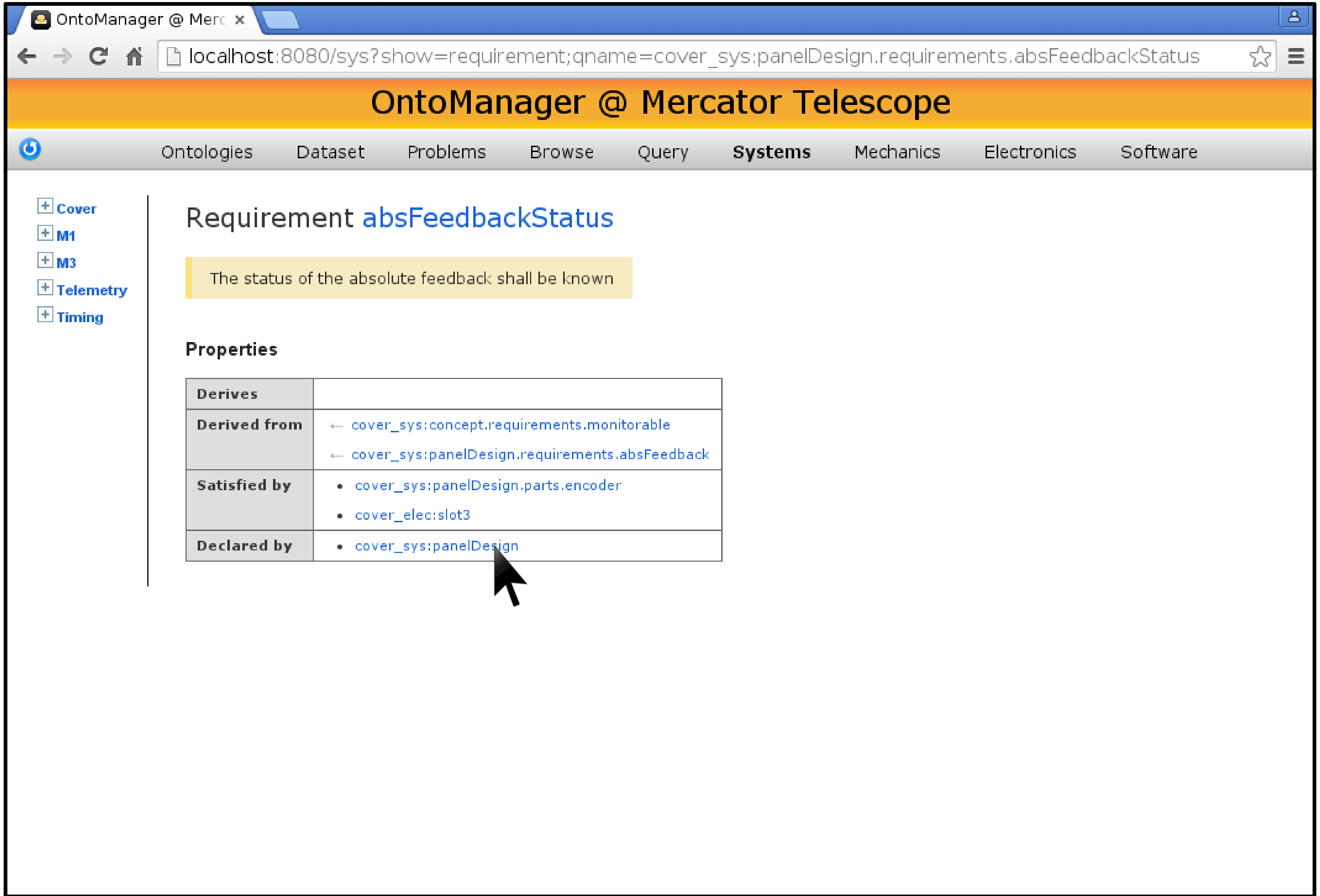
Properties

Derives	
Derived from	← cover_sys:concept.requirements.monitorable ← cover_sys:panelDesign.requirements.absFeedback
Satisfied by	• cover_sys:panelDesign.parts.encoder • cover_elec:slot3
Declared by	• cover_sys:panelDesign

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Systems design


OntoManager @ Merc
localhost:8080/sys?show=requirement;qname=cover_sys:panelDesign.requirements.absFeedbackStatus
OntoManager @ Mercator Telescope
Ontologies
Dataset
Problems
Browse
Query
Systems
Mechanics
Electronics
Software
Cover
M1
M3
Telemetry
Timing

Requirement absFeedbackStatus

The status of the absolute feedback shall be known

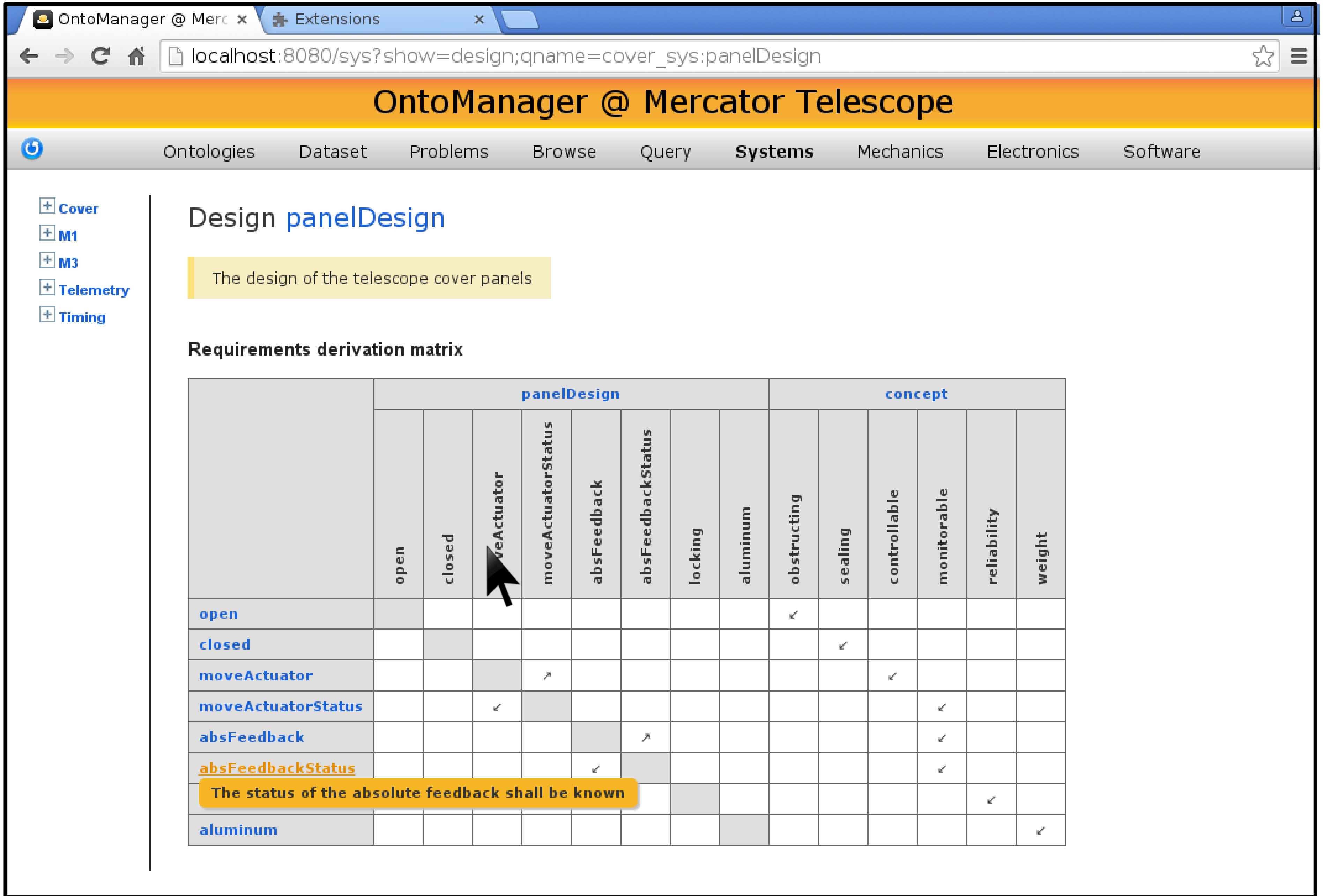
Properties

Derives	
Derived from	← cover_sys:concept.requirements.monitorable ← cover_sys:panelDesign.requirements.absFeedback
Satisfied by	• cover_sys:panelDesign.parts.encoder • cover_elec:slot3
Declared by	• cover_sys:panelDesign

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Systems design



Design panelDesign

The design of the telescope cover panels

Requirements derivation matrix

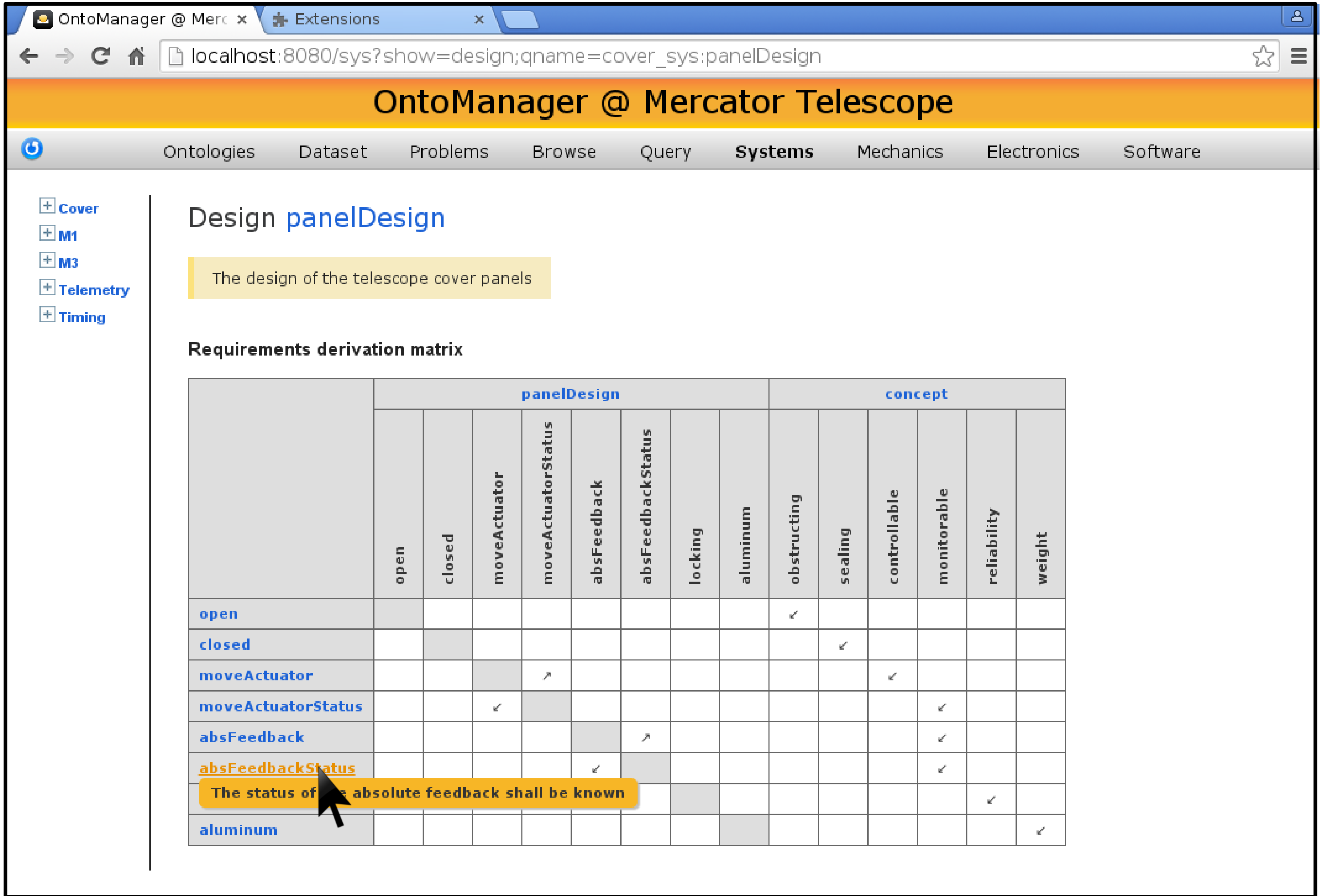
	panelDesign								concept					
	open	closed	veActuator	moveActuatorStatus	absFeedback	absFeedbackStatus	locking	aluminum	obstructing	sealing	controllable	monitorable	reliability	weight
open									↙					
closed										↙				
moveActuator				↗							↙			
moveActuatorStatus			↙									↙		
absFeedback						↗						↙		
absFeedbackStatus					↙							↙		
													↙	
aluminum														↙

The status of the absolute feedback shall be known

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Systems design



OntoManager @ Mercator Telescope

Design panelDesign

The design of the telescope cover panels

Requirements derivation matrix

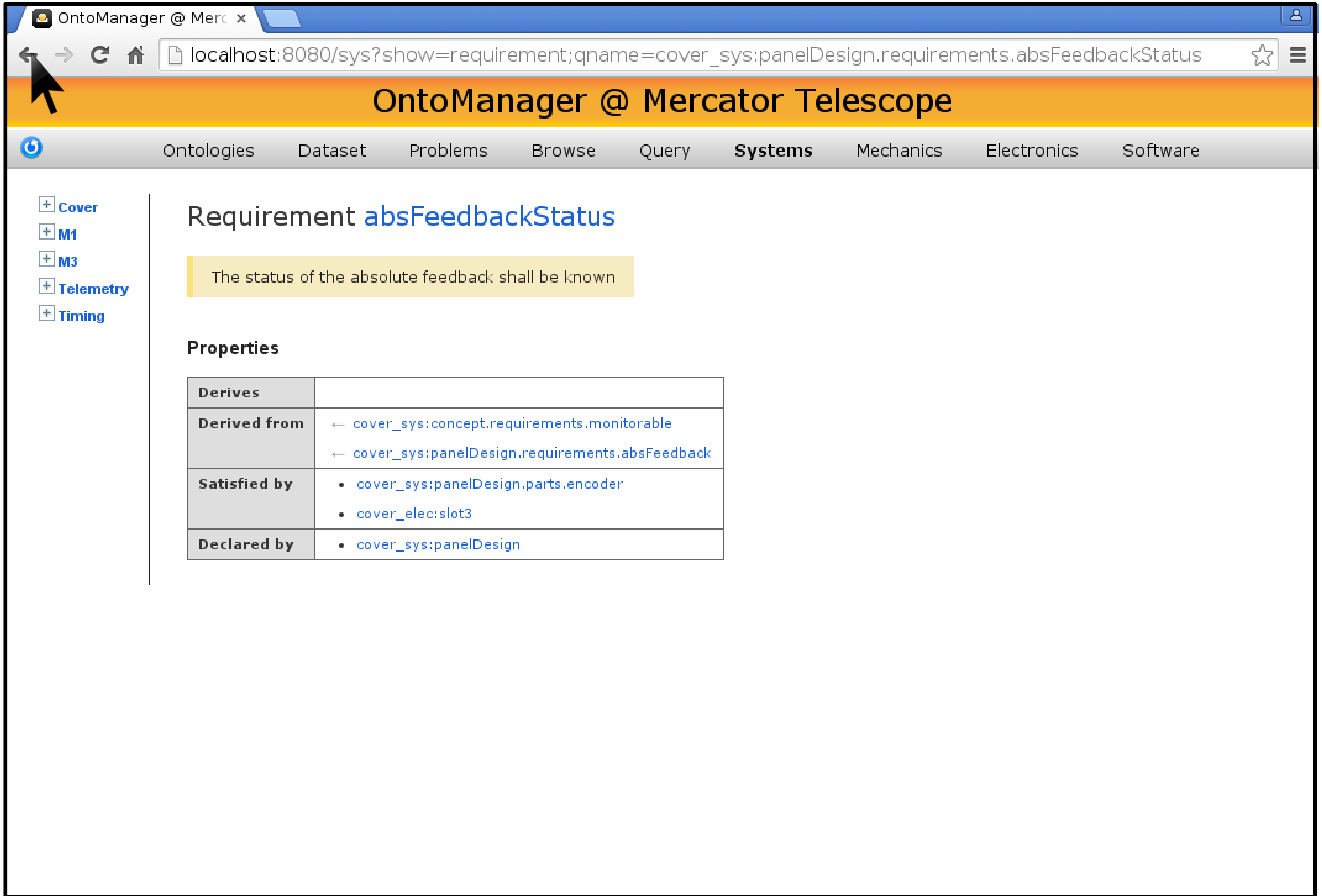
	panelDesign								concept					
	open	closed	moveActuator	moveActuatorStatus	absFeedback	absFeedbackStatus	locking	aluminum	obstructing	sealing	controllable	monitorable	reliability	weight
open									↙					
closed										↙				
moveActuator				↗							↙			
moveActuatorStatus			↙									↙		
absFeedback						↗						↙		
absFeedbackStatus					↙							↙		
[illegible]													↙	
aluminum														↙

The status of the absolute feedback shall be known

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Systems design



OntoManager @ Merc ×
localhost:8080/sys?show=requirement;qname=cover_sys:panelDesign.requirements.absFeedbackStatus

OntoManager @ Mercator Telescope

Ontologies | Dataset | Problems | Browse | Query | **Systems** | Mechanics | Electronics | Software

- Cover
- M1
- M3
- Telemetry
- Timing

Requirement absFeedbackStatus

The status of the absolute feedback shall be known

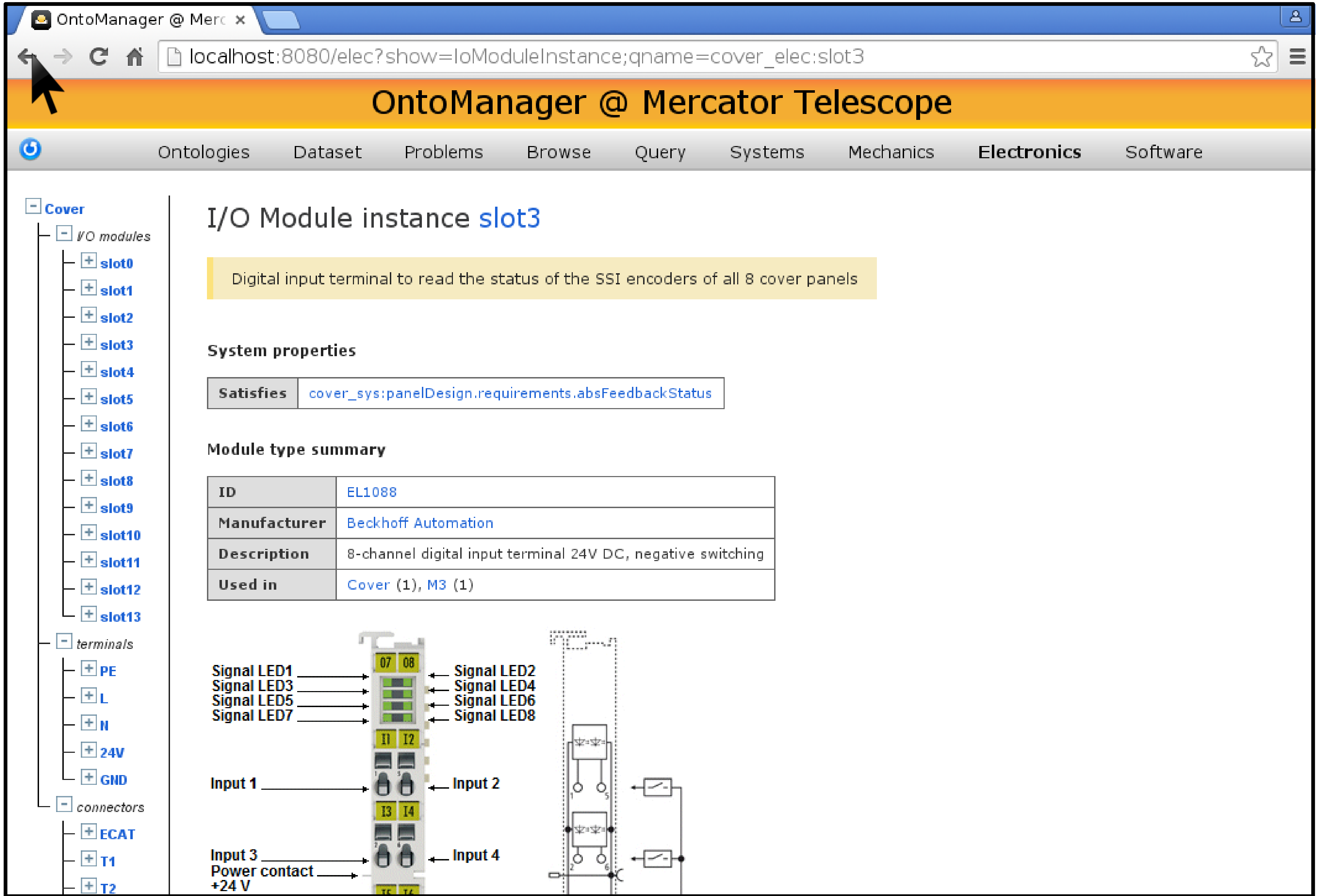
Properties

Derives	
Derived from	← cover_sys:concept.requirements.monitorable ← cover_sys:panelDesign.requirements.absFeedback
Satisfied by	• cover_sys:panelDesign.parts.encoder • cover_elec:slot3
Declared by	• cover_sys:panelDesign

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Electrical design



I/O Module instance slot3

Digital input terminal to read the status of the SSI encoders of all 8 cover panels

System properties

Satisfies	cover_sys:panelDesign.requirements.absFeedbackStatus

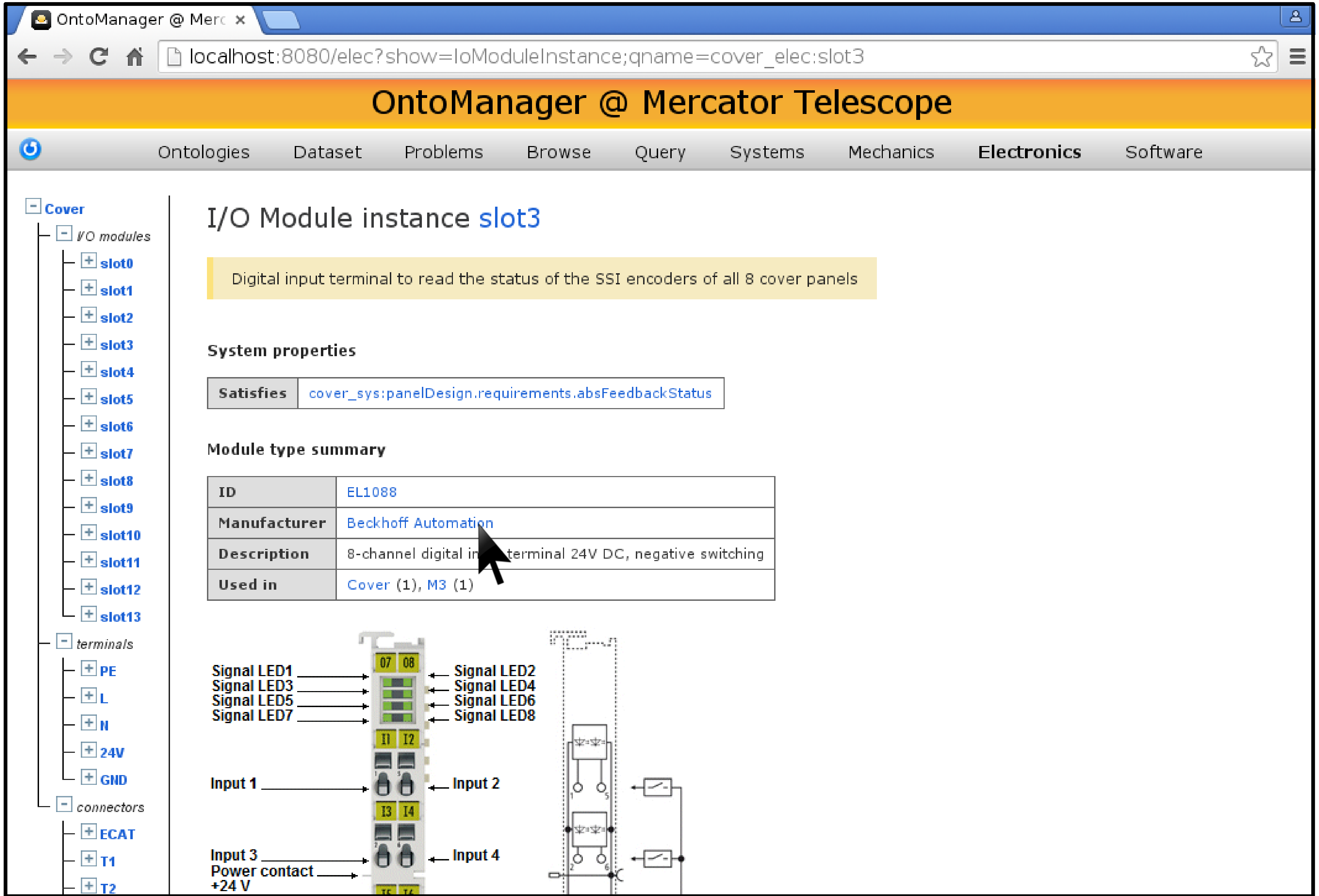
Module type summary

ID	EL1088
Manufacturer	Beckhoff Automation
Description	8-channel digital input terminal 24V DC, negative switching
Used in	Cover (1), M3 (1)

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Electrical design



OntoManager @ Merc ×
localhost:8080/elec?show=IoModuleInstance;qname=cover_elec:slot3
OntoManager @ Mercator Telescope
Ontologies
Dataset
Problems
Browse
Query
Systems
Mechanics
Electronics
Software

- Cover
  - I/O modules
    - slot0
    - slot1
    - slot2
    - slot3
    - slot4
    - slot5
    - slot6
    - slot7
    - slot8
    - slot9
    - slot10
    - slot11
    - slot12
    - slot13
  - terminals
    - PE
    - L
    - N
    - 24V
    - GND
  - connectors
    - ECAT
    - T1
    - T2

I/O Module instance slot3

Digital input terminal to read the status of the SSI encoders of all 8 cover panels

System properties

Satisfies	cover_sys:panelDesign.requirements.absFeedbackStatus

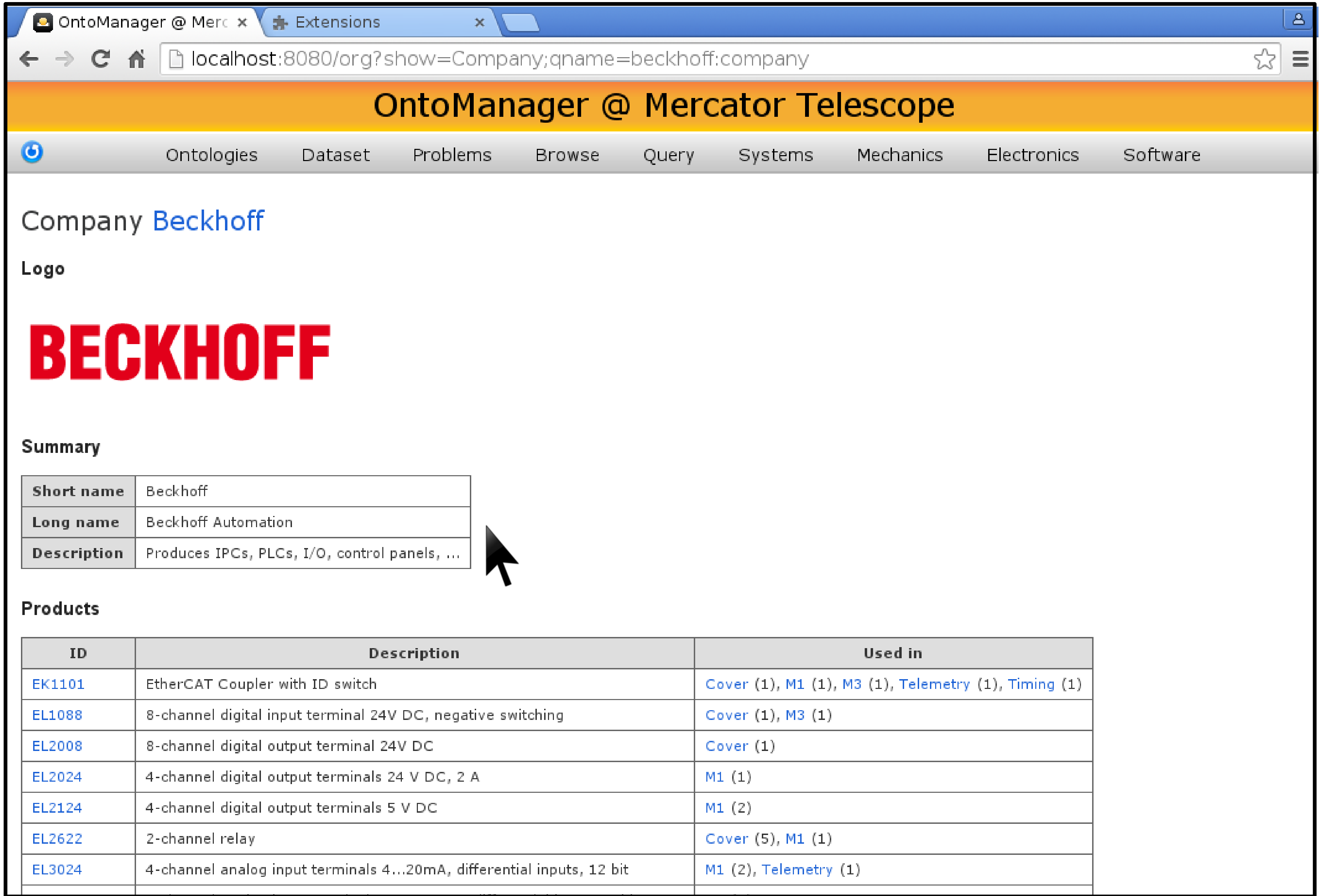
Module type summary

ID	EL1088
Manufacturer	Beckhoff Automation
Description	8-channel digital input terminal 24V DC, negative switching
Used in	Cover (1), M3 (1)

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Electrical design



Company Beckhoff

Logo

BECKHOFF

Summary

Short name	Beckhoff
Long name	Beckhoff Automation
Description	Produces IPCs, PLCs, I/O, control panels, ...

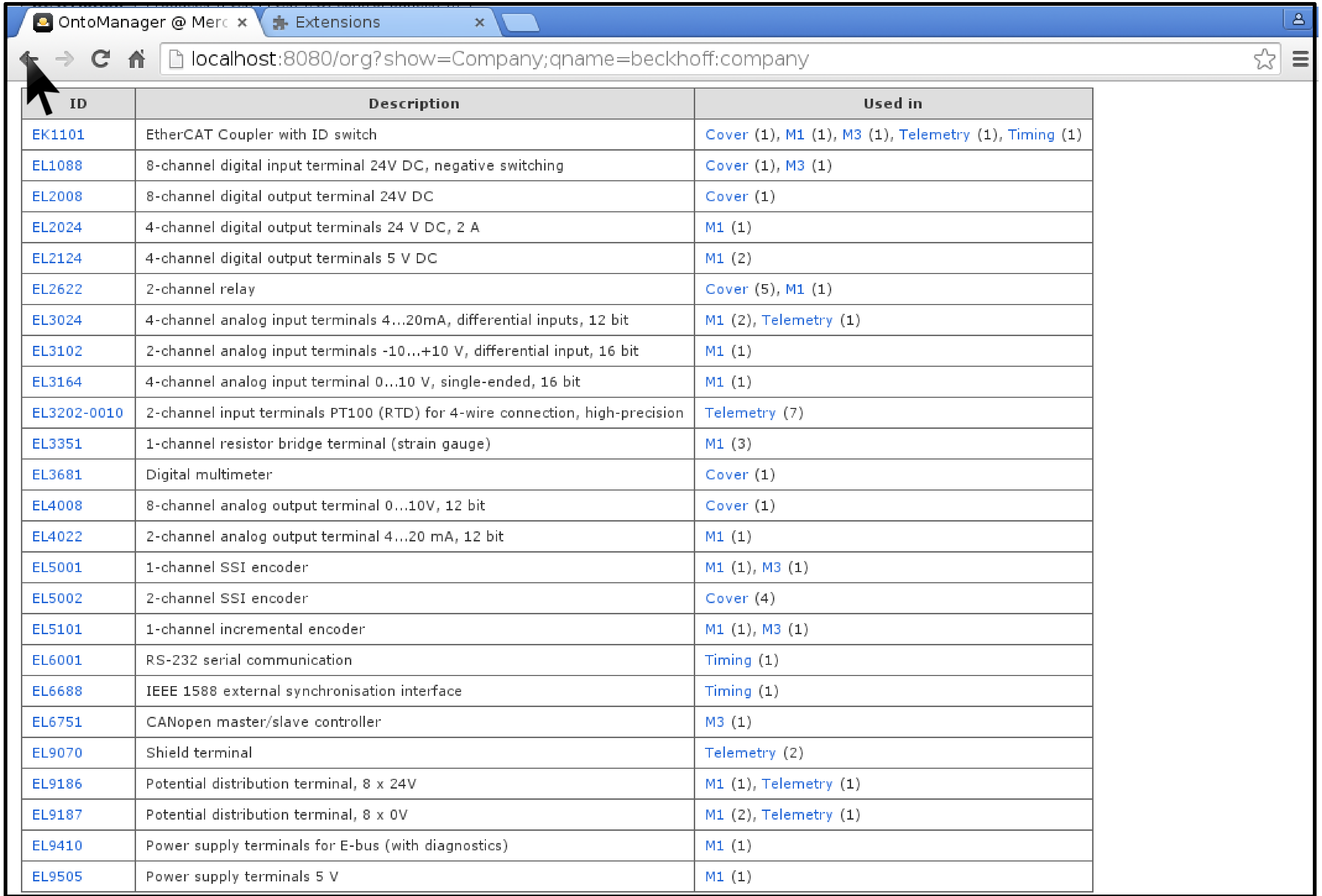
Products

ID	Description	Used in
EK1101	EtherCAT Coupler with ID switch	Cover (1), M1 (1), M3 (1), Telemetry (1), Timing (1)
EL1088	8-channel digital input terminal 24V DC, negative switching	Cover (1), M3 (1)
EL2008	8-channel digital output terminal 24V DC	Cover (1)
EL2024	4-channel digital output terminals 24 V DC, 2 A	M1 (1)
EL2124	4-channel digital output terminals 5 V DC	M1 (2)
EL2622	2-channel relay	Cover (5), M1 (1)
EL3024	4-channel analog input terminals 4...20mA, differential inputs, 12 bit	M1 (2), Telemetry (1)

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Electrical design

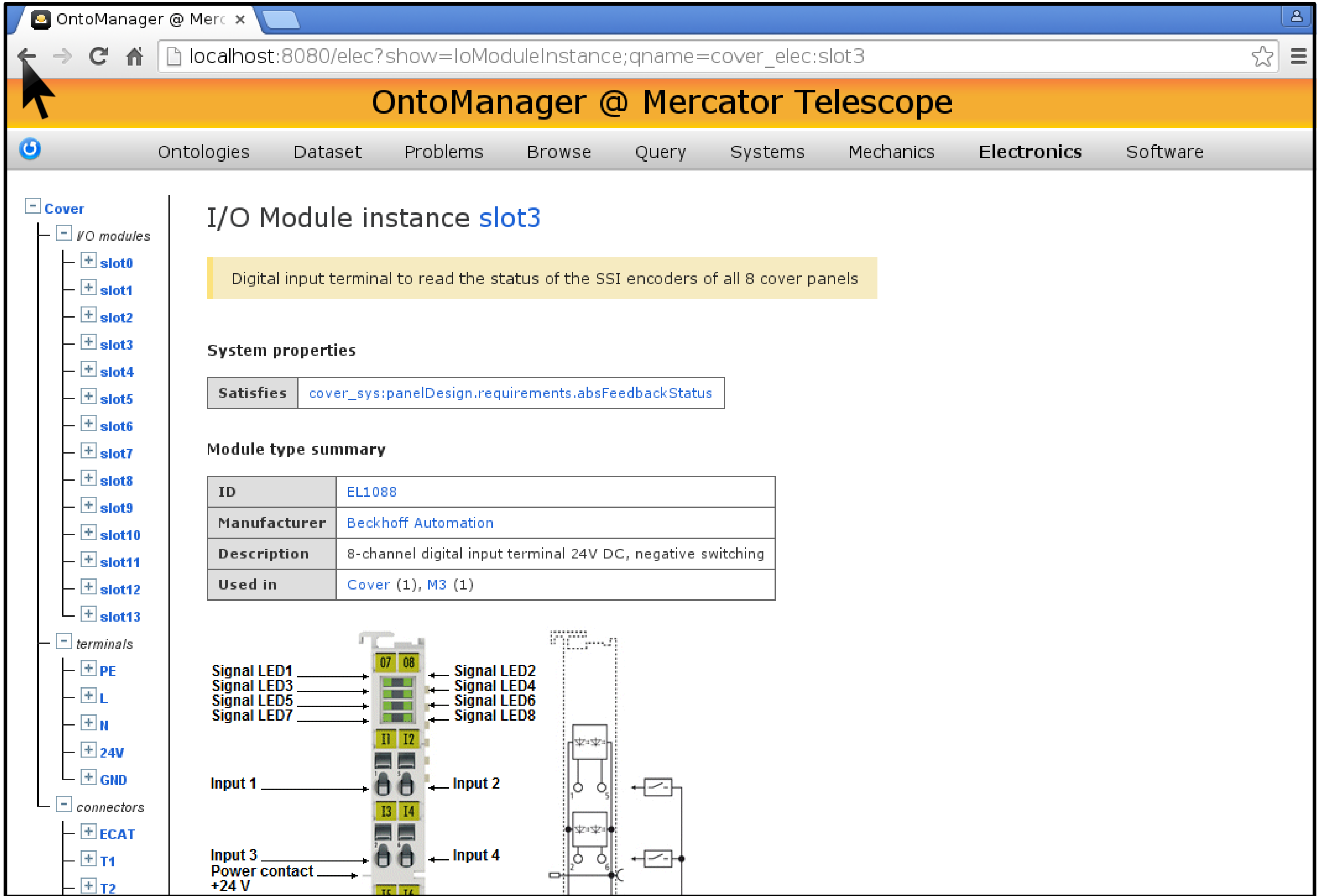


ID	Description	Used in
EK1101	EtherCAT Coupler with ID switch	Cover (1), M1 (1), M3 (1), Telemetry (1), Timing (1)
EL1088	8-channel digital input terminal 24V DC, negative switching	Cover (1), M3 (1)
EL2008	8-channel digital output terminal 24V DC	Cover (1)
EL2024	4-channel digital output terminals 24 V DC, 2 A	M1 (1)
EL2124	4-channel digital output terminals 5 V DC	M1 (2)
EL2622	2-channel relay	Cover (5), M1 (1)
EL3024	4-channel analog input terminals 4...20mA, differential inputs, 12 bit	M1 (2), Telemetry (1)
EL3102	2-channel analog input terminals -10...+10 V, differential input, 16 bit	M1 (1)
EL3164	4-channel analog input terminal 0...10 V, single-ended, 16 bit	M1 (1)
EL3202-0010	2-channel input terminals PT100 (RTD) for 4-wire connection, high-precision	Telemetry (7)
EL3351	1-channel resistor bridge terminal (strain gauge)	M1 (3)
EL3681	Digital multimeter	Cover (1)
EL4008	8-channel analog output terminal 0...10V, 12 bit	Cover (1)
EL4022	2-channel analog output terminal 4...20 mA, 12 bit	M1 (1)
EL5001	1-channel SSI encoder	M1 (1), M3 (1)
EL5002	2-channel SSI encoder	Cover (4)
EL5101	1-channel incremental encoder	M1 (1), M3 (1)
EL6001	RS-232 serial communication	Timing (1)
EL6688	IEEE 1588 external synchronisation interface	Timing (1)
EL6751	CANopen master/slave controller	M3 (1)
EL9070	Shield terminal	Telemetry (2)
EL9186	Potential distribution terminal, 8 x 24V	M1 (1), Telemetry (1)
EL9187	Potential distribution terminal, 8 x 0V	M1 (2), Telemetry (1)
EL9410	Power supply terminals for E-bus (with diagnostics)	M1 (1)
EL9505	Power supply terminals 5 V	M1 (1)

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Electrical design



I/O Module instance slot3

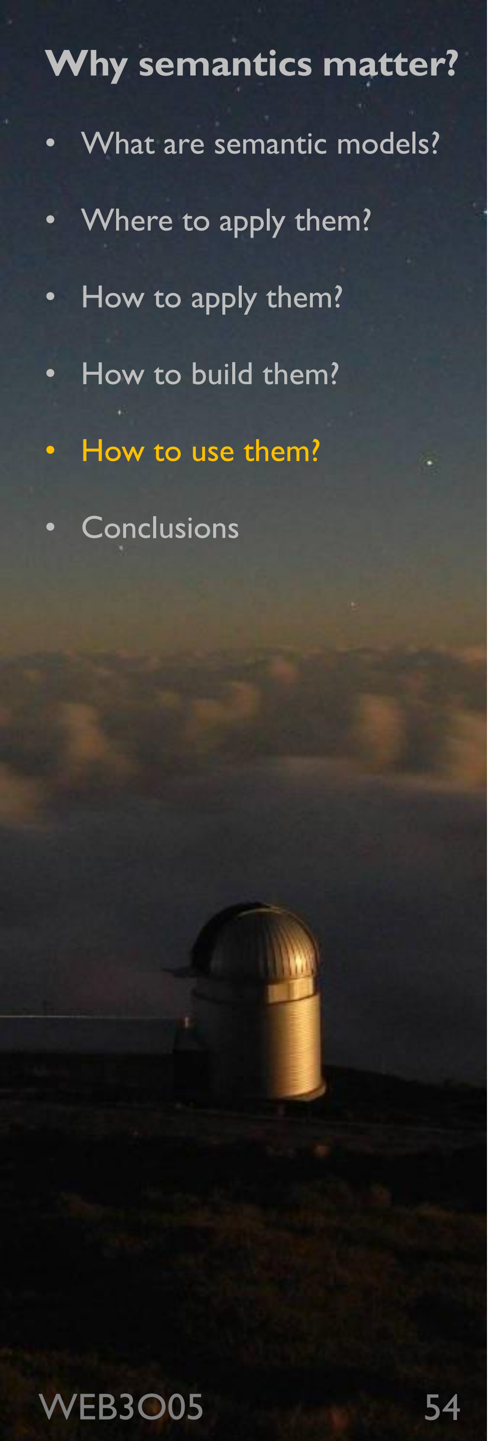
Digital input terminal to read the status of the SSI encoders of all 8 cover panels

System properties

Satisfies	cover_sys:panelDesign.requirements.absFeedbackStatus

Module type summary

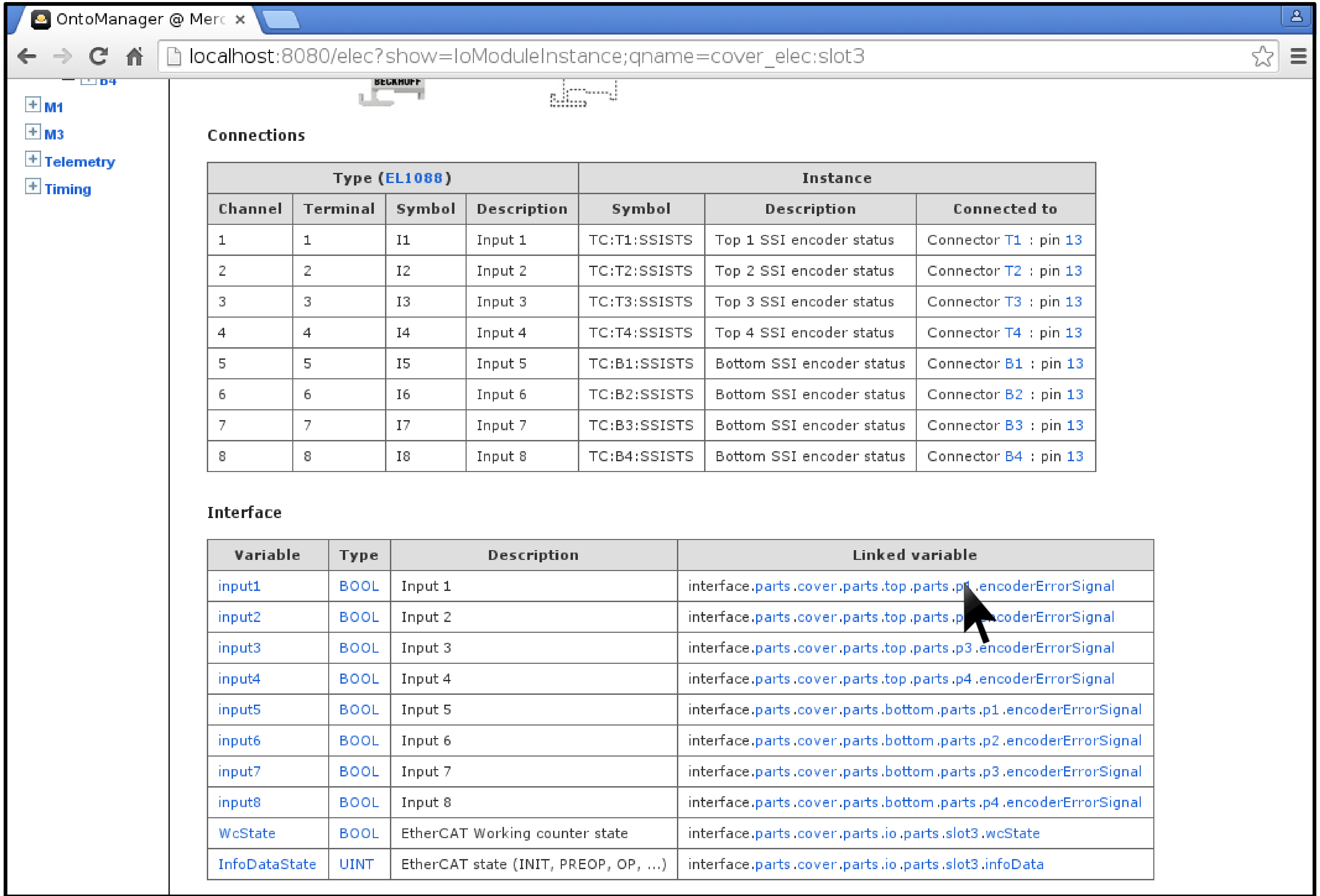
ID	EL1088
Manufacturer	Beckhoff Automation
Description	8-channel digital input terminal 24V DC, negative switching
Used in	Cover (1), M3 (1)



Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Electrical design

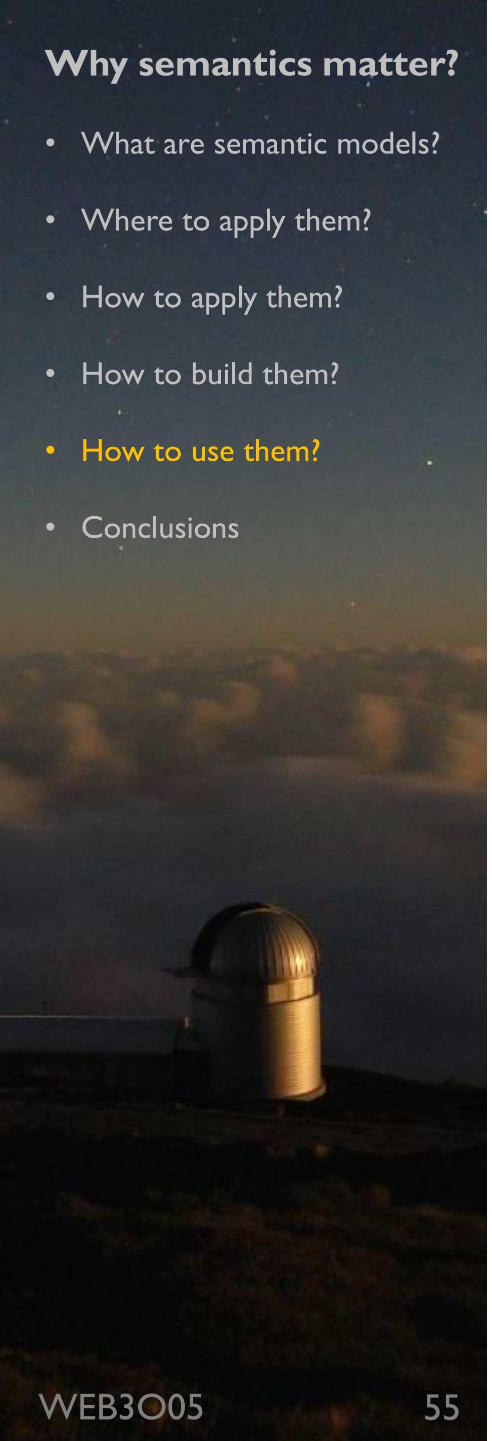

OntoManager @ Merc
localhost:8080/elec?show=IoModuleInstance;qname=cover_elec:slot3
B4
M1
M3
Telemetry
Timing
BECKHOFF

Connections

Type (EL1088)				Instance		
Channel	Terminal	Symbol	Description	Symbol	Description	Connected to
1	1	I1	Input 1	TC:T1:SSISTS	Top 1 SSI encoder status	Connector T1 : pin 13
2	2	I2	Input 2	TC:T2:SSISTS	Top 2 SSI encoder status	Connector T2 : pin 13
3	3	I3	Input 3	TC:T3:SSISTS	Top 3 SSI encoder status	Connector T3 : pin 13
4	4	I4	Input 4	TC:T4:SSISTS	Top 4 SSI encoder status	Connector T4 : pin 13
5	5	I5	Input 5	TC:B1:SSISTS	Bottom SSI encoder status	Connector B1 : pin 13
6	6	I6	Input 6	TC:B2:SSISTS	Bottom SSI encoder status	Connector B2 : pin 13
7	7	I7	Input 7	TC:B3:SSISTS	Bottom SSI encoder status	Connector B3 : pin 13
8	8	I8	Input 8	TC:B4:SSISTS	Bottom SSI encoder status	Connector B4 : pin 13

Interface

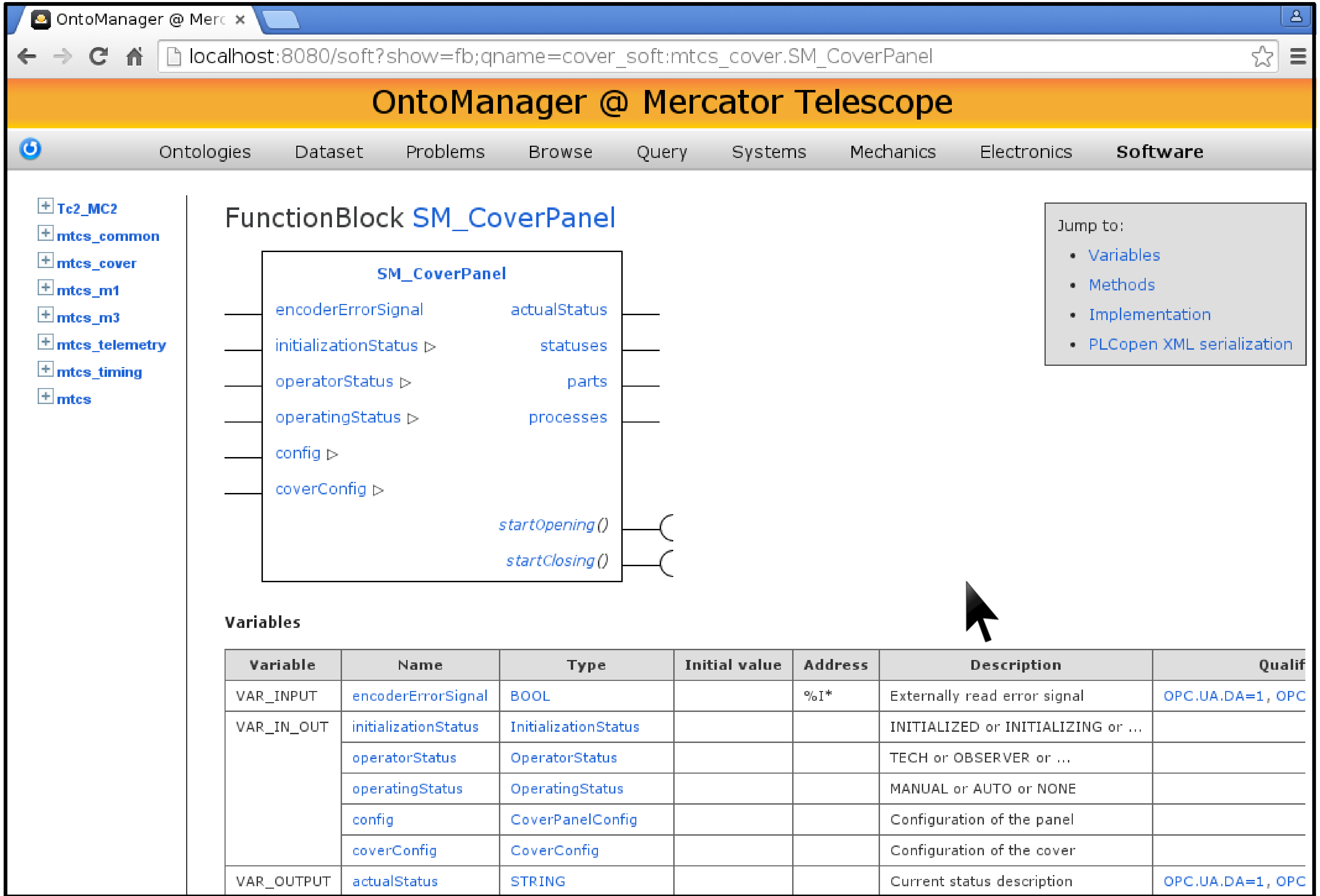
Variable	Type	Description	Linked variable
input1	BOOL	Input 1	interface.parts.cover.parts.top.parts.p1.encoderErrorSignal
input2	BOOL	Input 2	interface.parts.cover.parts.top.parts.p2.encoderErrorSignal
input3	BOOL	Input 3	interface.parts.cover.parts.top.parts.p3.encoderErrorSignal
input4	BOOL	Input 4	interface.parts.cover.parts.top.parts.p4.encoderErrorSignal
input5	BOOL	Input 5	interface.parts.cover.parts.bottom.parts.p1.encoderErrorSignal
input6	BOOL	Input 6	interface.parts.cover.parts.bottom.parts.p2.encoderErrorSignal
input7	BOOL	Input 7	interface.parts.cover.parts.bottom.parts.p3.encoderErrorSignal
input8	BOOL	Input 8	interface.parts.cover.parts.bottom.parts.p4.encoderErrorSignal
WcState	BOOL	EtherCAT Working counter state	interface.parts.cover.parts.io.parts.slot3.wcState
InfoDataState	UINT	EtherCAT state (INIT, PREOP, OP, ...)	interface.parts.cover.parts.io.parts.slot3.infoData



Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Software design



OntoManager @ Merc ×
localhost:8080/soft?show=fb;qname=cover_soft:mtcs_cover.SM_CoverPanel

OntoManager @ Mercator Telescope

FunctionBlock SM_CoverPanel

Jump to:

- Variables
- Methods
- Implementation
- PLCopen XML serialization

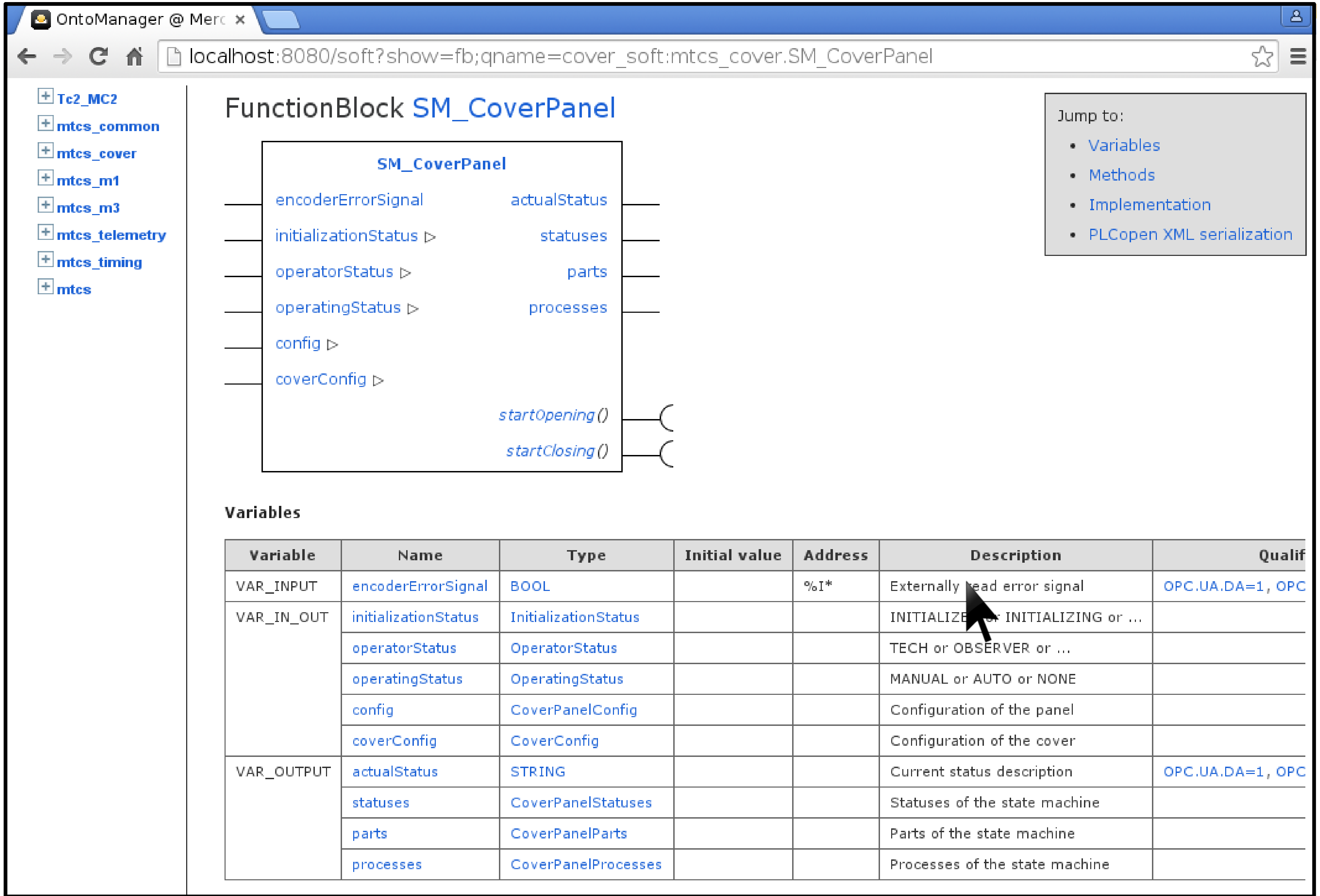
Variables

Variable	Name	Type	Initial value	Address	Description	Qualif
VAR_INPUT	encoderErrorSignal	BOOL		%I*	Externally read error signal	OPC.UA.DA=1, OPC
VAR_IN_OUT	initializationStatus	InitializationStatus			INITIALIZED or INITIALIZING or ...	
	operatorStatus	OperatorStatus			TECH or OBSERVER or ...	
	operatingStatus	OperatingStatus			MANUAL or AUTO or NONE	
	config	CoverPanelConfig			Configuration of the panel	
	coverConfig	CoverConfig			Configuration of the cover	
VAR_OUTPUT	actualStatus	STRING			Current status description	OPC.UA.DA=1, OPC

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Software design



OntoManager @ Merc ×

localhost:8080/soft?show=fb;qname=cover_soft:mtcs_cover.SM_CoverPanel

- Tc2_MC2
- mtcs_common
- mtcs_cover
- mtcs_m1
- mtcs_m3
- mtcs_telemetry
- mtcs_timing
- mtcs

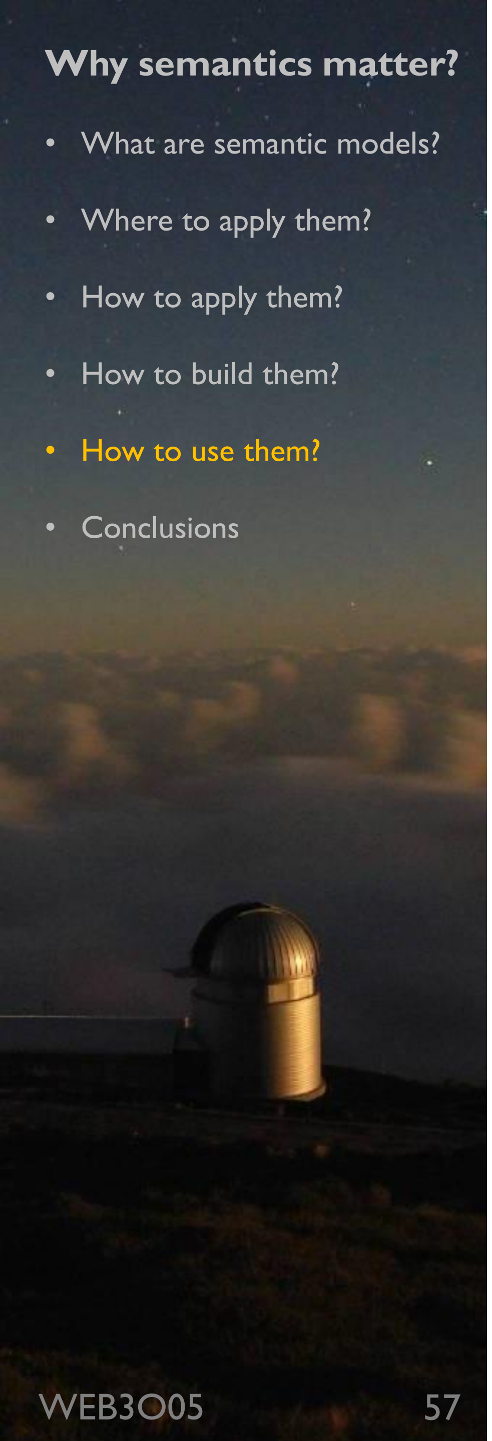
FunctionBlock SM_CoverPanel

Jump to:

- Variables
- Methods
- Implementation
- PLCopen XML serialization

Variables

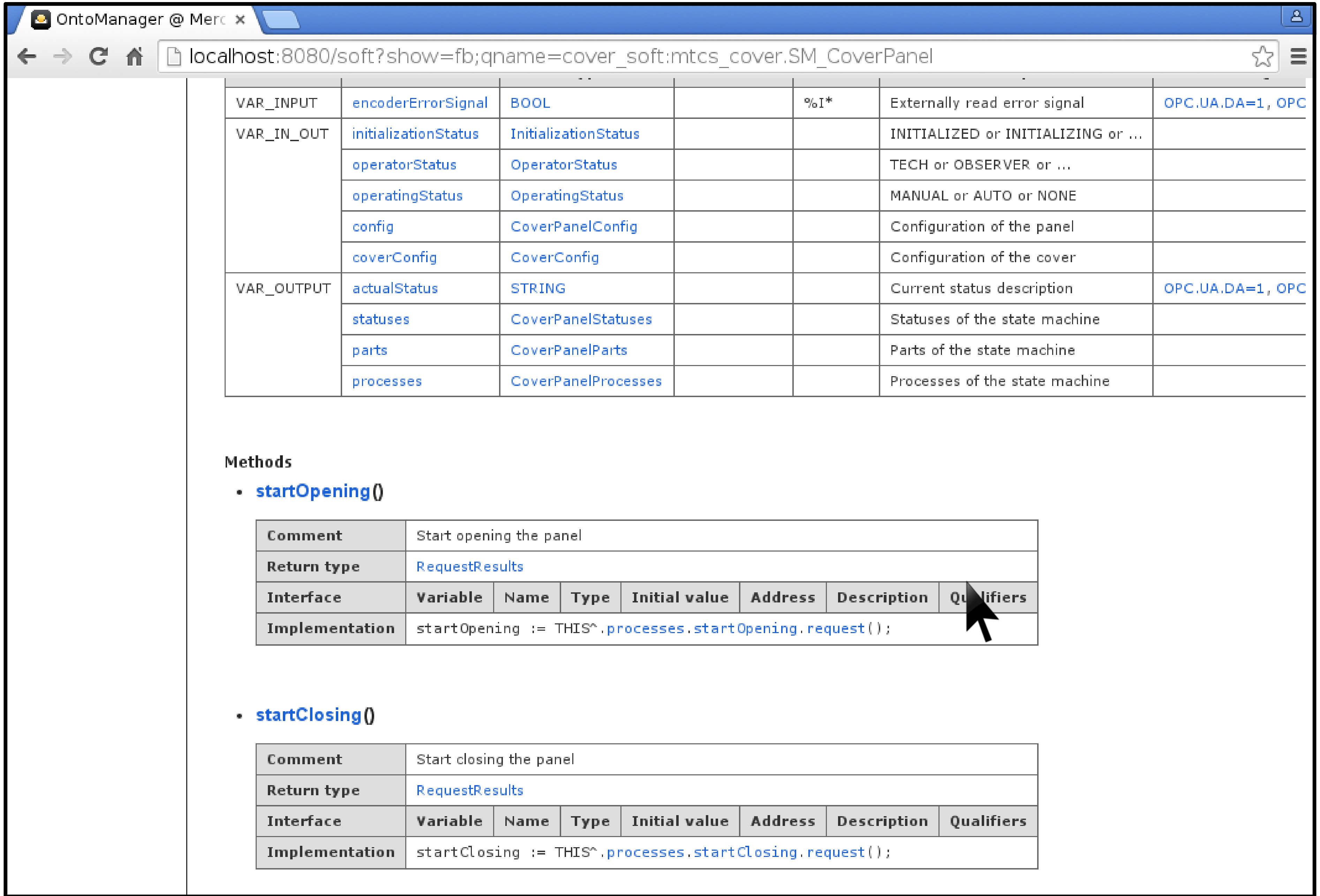
Variable	Name	Type	Initial value	Address	Description	Qualif
VAR_INPUT	encoderErrorSignal	BOOL		%I*	Externally read error signal	OPC.UA.DA=1, OPC
VAR_IN_OUT	initializationStatus	InitializationStatus			INITIALIZED or INITIALIZING or ...	
	operatorStatus	OperatorStatus			TECH or OBSERVER or ...	
	operatingStatus	OperatingStatus			MANUAL or AUTO or NONE	
	config	CoverPanelConfig			Configuration of the panel	
	coverConfig	CoverConfig			Configuration of the cover	
VAR_OUTPUT	actualStatus	STRING			Current status description	OPC.UA.DA=1, OPC
	statuses	CoverPanelStatuses			Statuses of the state machine	
	parts	CoverPanelParts			Parts of the state machine	
	processes	CoverPanelProcesses			Processes of the state machine	



Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Software design



OntoManager @ Merc ×
localhost:8080/soft?show=fb;qname=cover_soft:mtcs_cover.SM_CoverPanel

VAR_INPUT	encoderErrorSignal	BOOL		%I*	Externally read error signal	OPC.UA.DA=1, OPC
VAR_IN_OUT	initializationStatus	InitializationStatus			INITIALIZED or INITIALIZING or ...	
	operatorStatus	OperatorStatus			TECH or OBSERVER or ...	
	operatingStatus	OperatingStatus			MANUAL or AUTO or NONE	
	config	CoverPanelConfig			Configuration of the panel	
	coverConfig	CoverConfig			Configuration of the cover	
VAR_OUTPUT	actualStatus	STRING			Current status description	OPC.UA.DA=1, OPC
	statuses	CoverPanelStatuses			Statuses of the state machine	
	parts	CoverPanelParts			Parts of the state machine	
	processes	CoverPanelProcesses			Processes of the state machine	

Methods

- **startOpening()**

Comment	Start opening the panel						
Return type	RequestResults						
Interface	**Variable**	**Name**	**Type**	**Initial value**	**Address**	**Description**	**Qualifiers**
Implementation	`startOpening := THIS^.processes.startOpening.request();`						

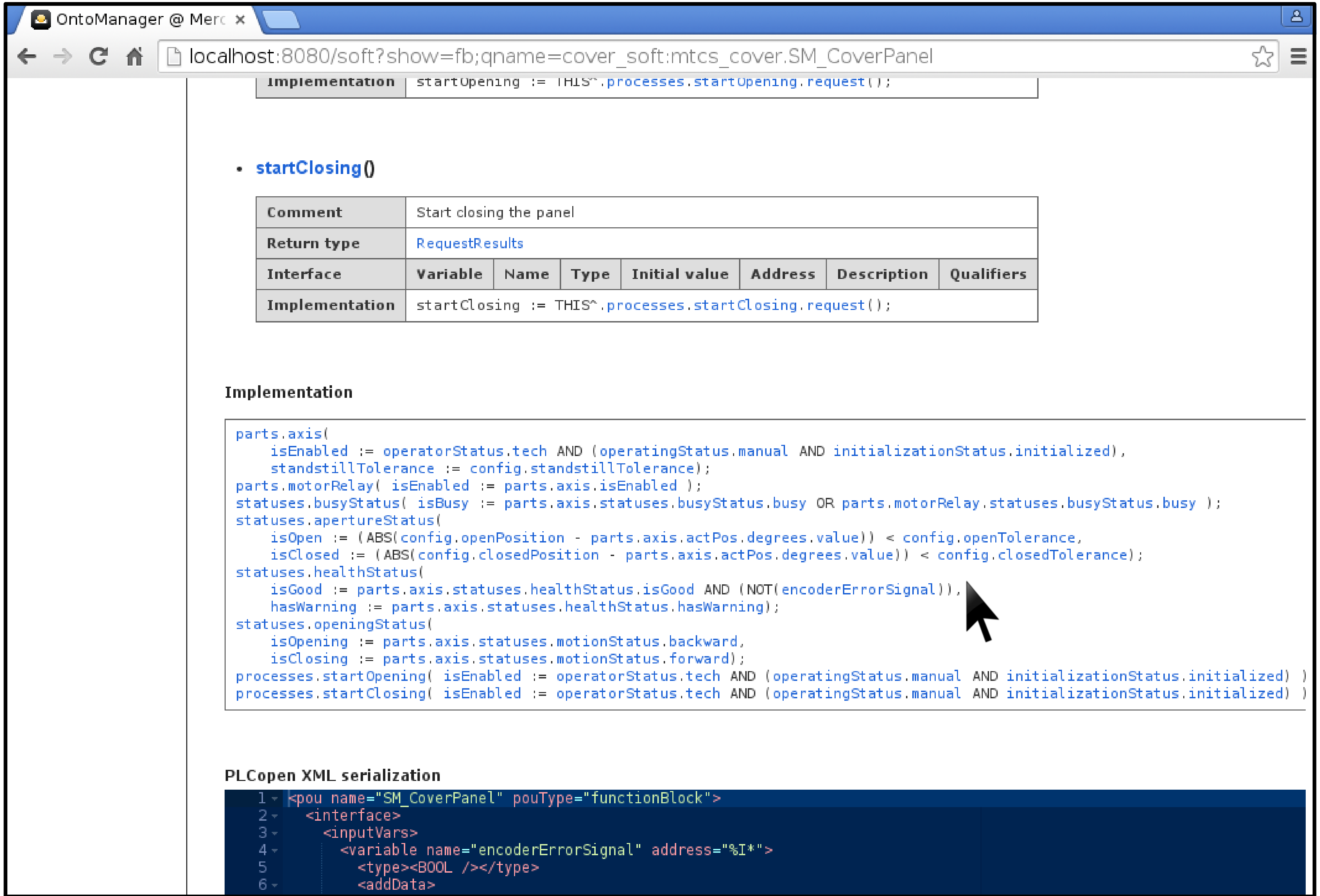
- **startClosing()**

Comment	Start closing the panel						
Return type	RequestResults						
Interface	**Variable**	**Name**	**Type**	**Initial value**	**Address**	**Description**	**Qualifiers**
Implementation	`startClosing := THIS^.processes.startClosing.request();`						

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Software design



OntoManager @ Merc ×
localhost:8080/soft?show=fb;qname=cover_soft:mtcs_cover.SM_CoverPanel

Implementation	startOpening := THIS^.processes.startOpening.request();

- **startClosing()**

Comment	Start closing the panel						
Return type	RequestResults						
Interface	**Variable**	**Name**	**Type**	**Initial value**	**Address**	**Description**	**Qualifiers**
Implementation	startClosing := THIS^.processes.startClosing.request();						

Implementation

```
parts.axis(
    isEnabled := operatorStatus.tech AND (operatingStatus.manual AND initializationStatus.initialized),
    standstillTolerance := config.standstillTolerance);
parts.motorRelay( isEnabled := parts.axis.isEnabled );
statuses.busyStatus( isBusy := parts.axis.statuses.busyStatus.busy OR parts.motorRelay.statuses.busyStatus.busy );
statuses.apertureStatus(
    isOpen := (ABS(config.openPosition - parts.axis.actPos.degrees.value)) < config.openTolerance,
    isClosed := (ABS(config.closedPosition - parts.axis.actPos.degrees.value)) < config.closedTolerance);
statuses.healthStatus(
    isGood := parts.axis.statuses.healthStatus.isGood AND (NOT(encoderErrorSignal)),
    hasWarning := parts.axis.statuses.healthStatus.hasWarning);
statuses.openingStatus(
    isOpening := parts.axis.statuses.motionStatus.backward,
    isClosing := parts.axis.statuses.motionStatus.forward);
processes.startOpening( isEnabled := operatorStatus.tech AND (operatingStatus.manual AND initializationStatus.initialized) )
processes.startClosing( isEnabled := operatorStatus.tech AND (operatingStatus.manual AND initializationStatus.initialized) )
```

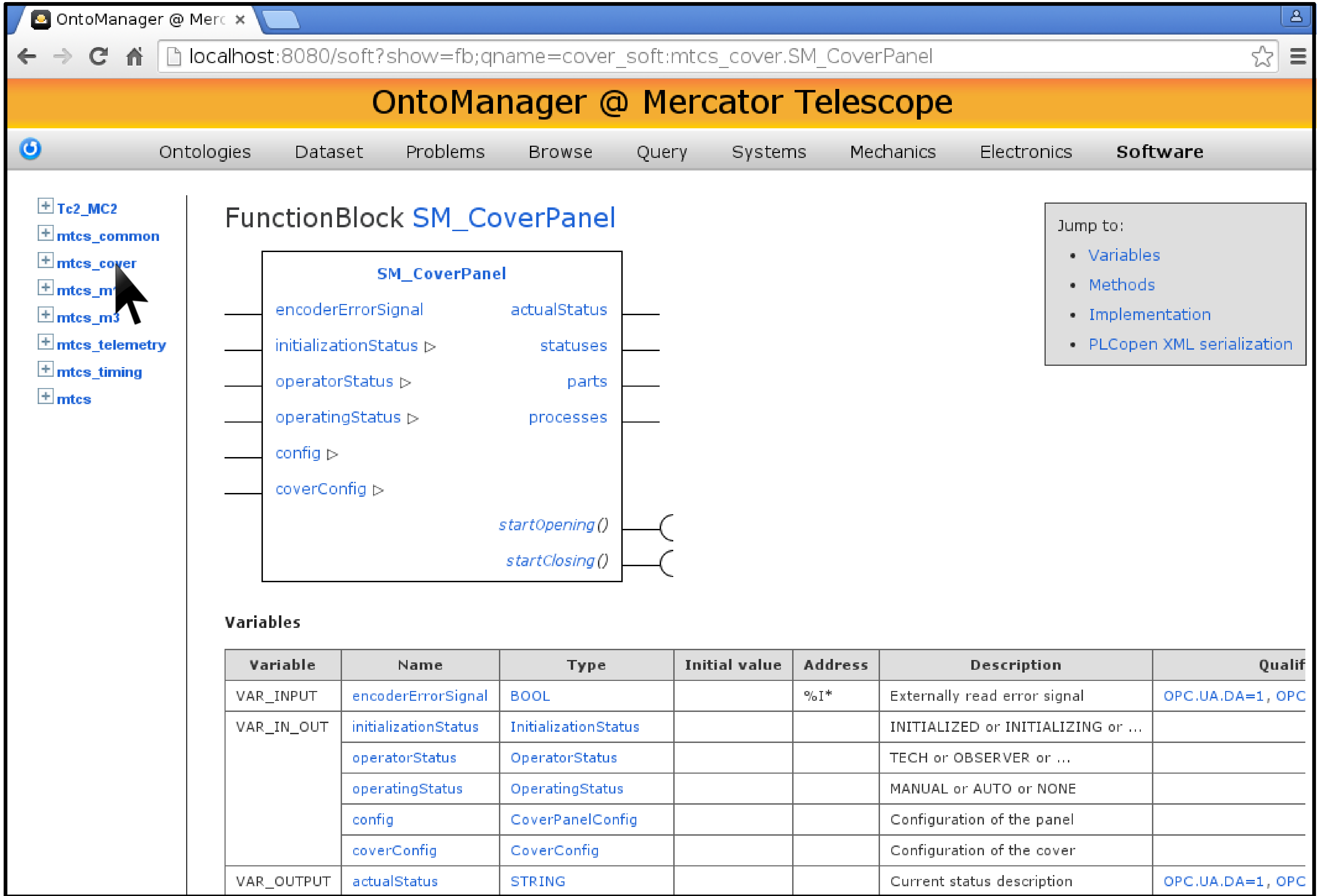
PLCopen XML serialization

```
<pou name="SM_CoverPanel" pouType="functionBlock">
  <interface>
    <inputVars>
      <variable name="encoderErrorSignal" address="%I*">
        <type><BOOL /></type>
        <addData>
```

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Software design



OntoManager @ Merc
localhost:8080/soft?show=fb;qname=cover_soft:mtcs_cover.SM_CoverPanel

OntoManager @ Mercator Telescope

FunctionBlock SM_CoverPanel

Jump to:

- Variables
- Methods
- Implementation
- PLCopen XML serialization

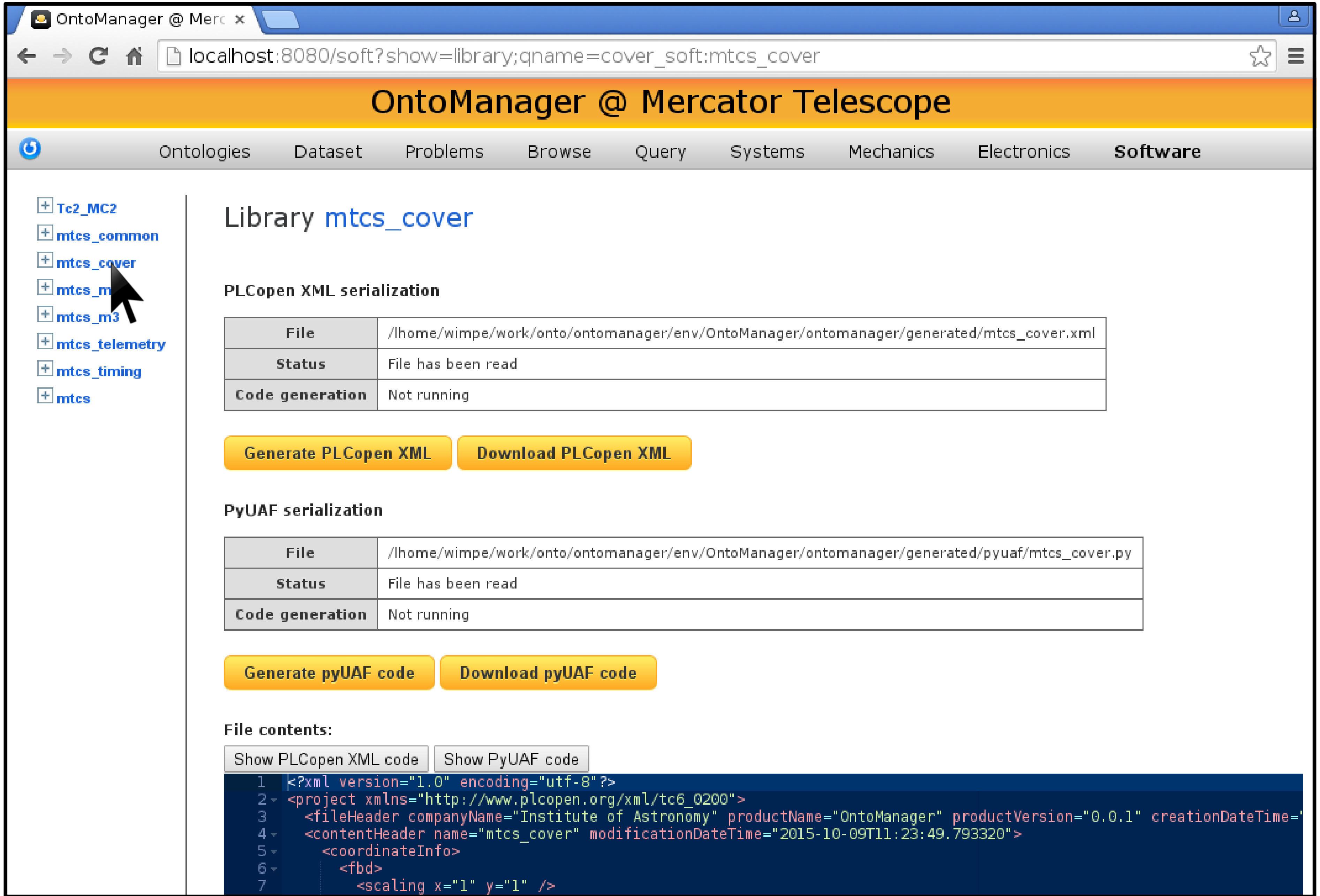
Variables

Variable	Name	Type	Initial value	Address	Description	Qualif
VAR_INPUT	encoderErrorSignal	BOOL		%I*	Externally read error signal	OPC.UA.DA=1, OPC
VAR_IN_OUT	initializationStatus	InitializationStatus			INITIALIZED or INITIALIZING or ...	
	operatorStatus	OperatorStatus			TECH or OBSERVER or ...	
	operatingStatus	OperatingStatus			MANUAL or AUTO or NONE	
	config	CoverPanelConfig			Configuration of the panel	
	coverConfig	CoverConfig			Configuration of the cover	
VAR_OUTPUT	actualStatus	STRING			Current status description	OPC.UA.DA=1, OPC

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Software design



OntoManager @ Merc ×
localhost:8080/soft?show=library;qname=cover_soft:mtcs_cover

OntoManager @ Mercator Telescope

Library mtcs_cover

PLCopen XML serialization

File	/lhome/wimpe/work/onto/ontomanager/env/OntoManager/ontomanager/generated/mtcs_cover.xml
Status	File has been read
Code generation	Not running

PyUAF serialization

File	/lhome/wimpe/work/onto/ontomanager/env/OntoManager/ontomanager/generated/pyuaf/mtcs_cover.py
Status	File has been read
Code generation	Not running

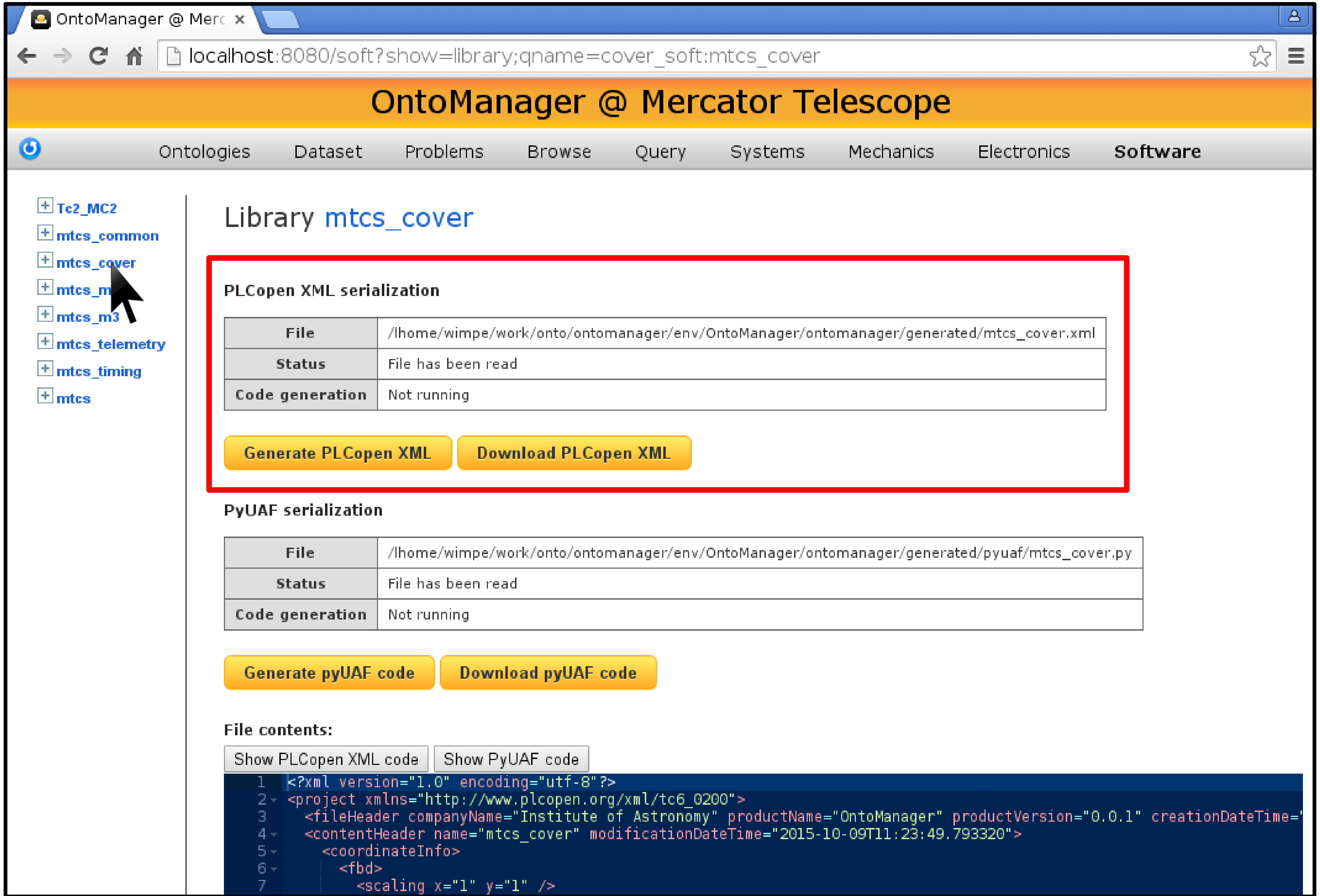
File contents:

```
<?xml version="1.0" encoding="utf-8"?>
<project xmlns="http://www.plcopen.org/xml/tc6_0200">
  <fileHeader companyName="Institute of Astronomy" productName="OntoManager" productVersion="0.0.1" creationDateTime="
  <contentHeader name="mtcs_cover" modificationDateTime="2015-10-09T11:23:49.793320">
    <coordinateInfo>
      <fbd>
        <scaling x="1" y="1" />
```

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Software design



Library mtcs_cover

PLCopen XML serialization

File	/lhome/wimpe/work/onto/ontomanager/env/OntoManager/ontomanager/generated/mtcs_cover.xml
Status	File has been read
Code generation	Not running

Generate PLCopen XML | Download PLCopen XML

PyUAF serialization

File	/lhome/wimpe/work/onto/ontomanager/env/OntoManager/ontomanager/generated/pyuaf/mtcs_cover.py
Status	File has been read
Code generation	Not running

Generate pyUAF code | Download pyUAF code

File contents:

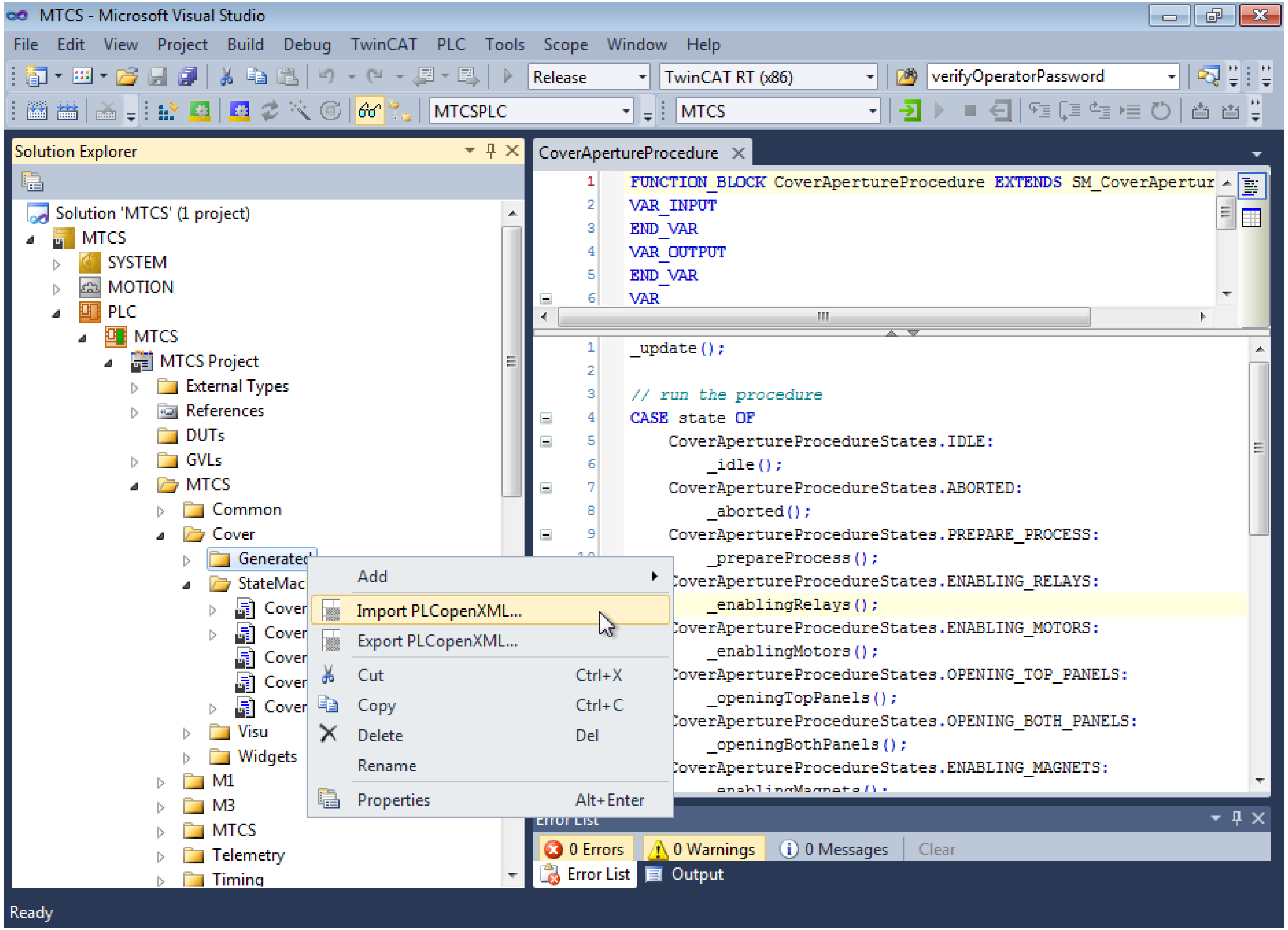
Show PLCopen XML code | Show PyUAF code

```
<?xml version="1.0" encoding="utf-8"?>
<project xmlns="http://www.plcopen.org/xml/tc6_0200">
  <fileHeader companyName="Institute of Astronomy" productName="OntoManager" productVersion="0.0.1" creationDateTime="
  <contentHeader name="mtcs_cover" modificationDateTime="2015-10-09T11:23:49.793320">
    <coordinateInfo>
      <fbd>
        <scaling x="1" y="1" />
```

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Software design



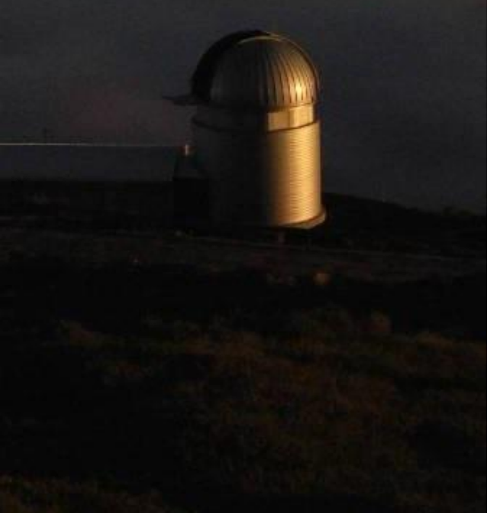
```
FUNCTION_BLOCK CoverApertureProcedure EXTENDS SM_CoverApertur
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
```

```
_update();

// run the procedure
CASE state OF
    CoverApertureProcedureStates.IDLE:
        _idle();
    CoverApertureProcedureStates.ABORTED:
        _aborted();
    CoverApertureProcedureStates.PREPARE_PROCESS:
        _prepareProcess();
    overApertureProcedureStates.ENABLING_RELAYS:
        _enablingRelays();
    overApertureProcedureStates.ENABLING_MOTORS:
        _enablingMotors();
    overApertureProcedureStates.OPENING_TOP_PANELS:
        _openingTopPanels();
    overApertureProcedureStates.OPENING_BOTH_PANELS:
        _openingBothPanels();
    overApertureProcedureStates.ENABLING_MAGNETS:
        _enablingMagnets();
```

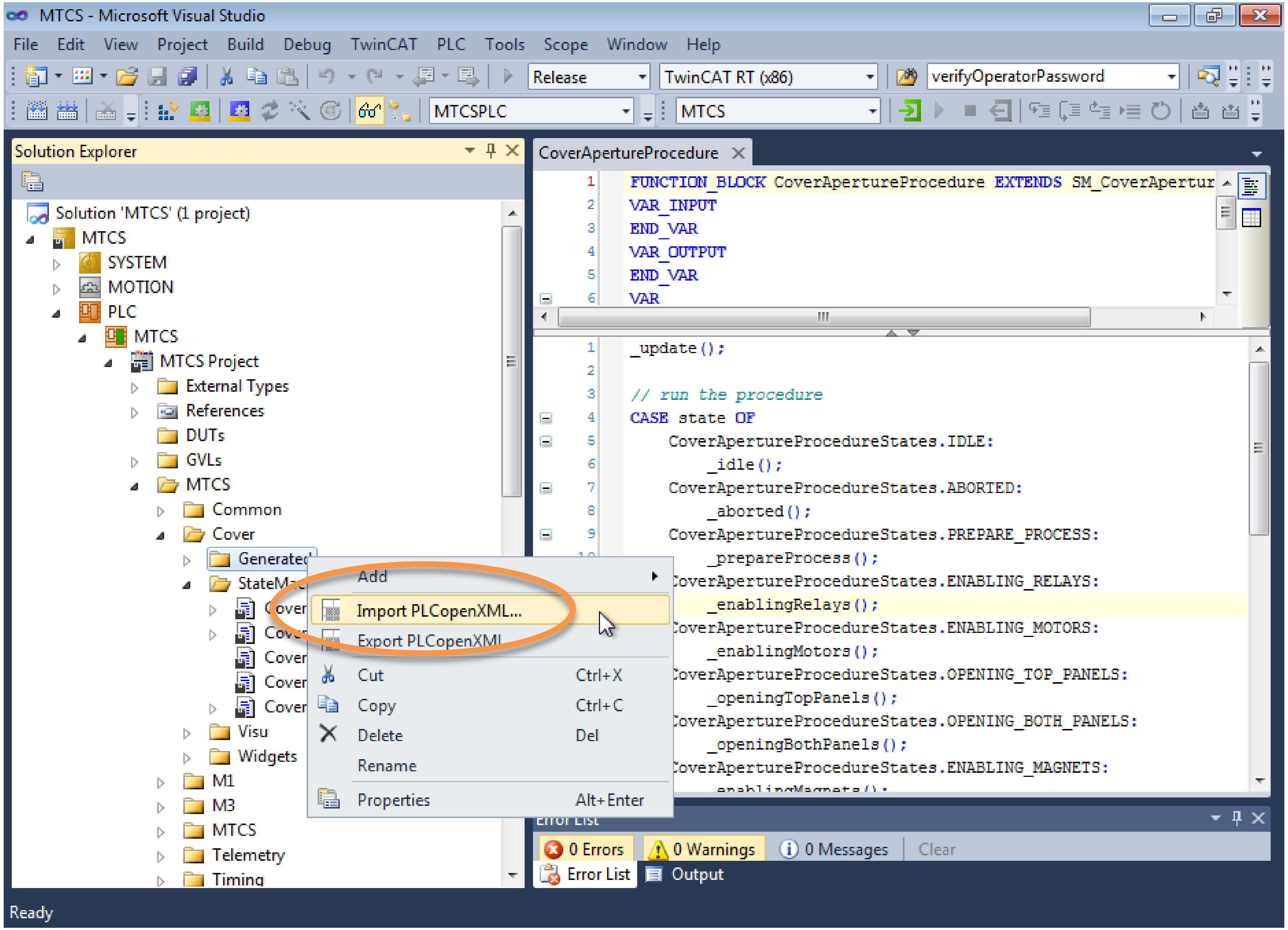
0 Errors | 0 Warnings | 0 Messages

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions



Software design

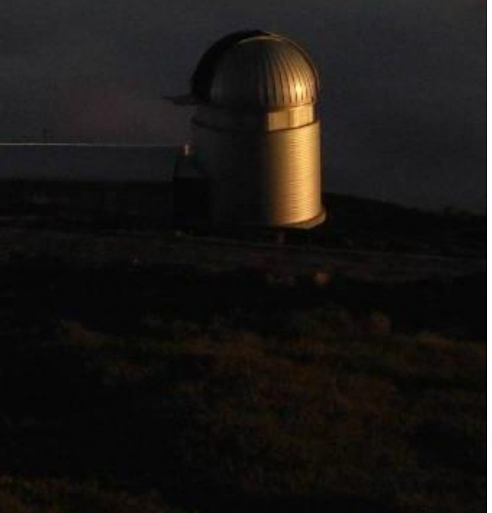


```
FUNCTION_BLOCK CoverApertureProcedure EXTENDS SM_CoverApertur
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
```

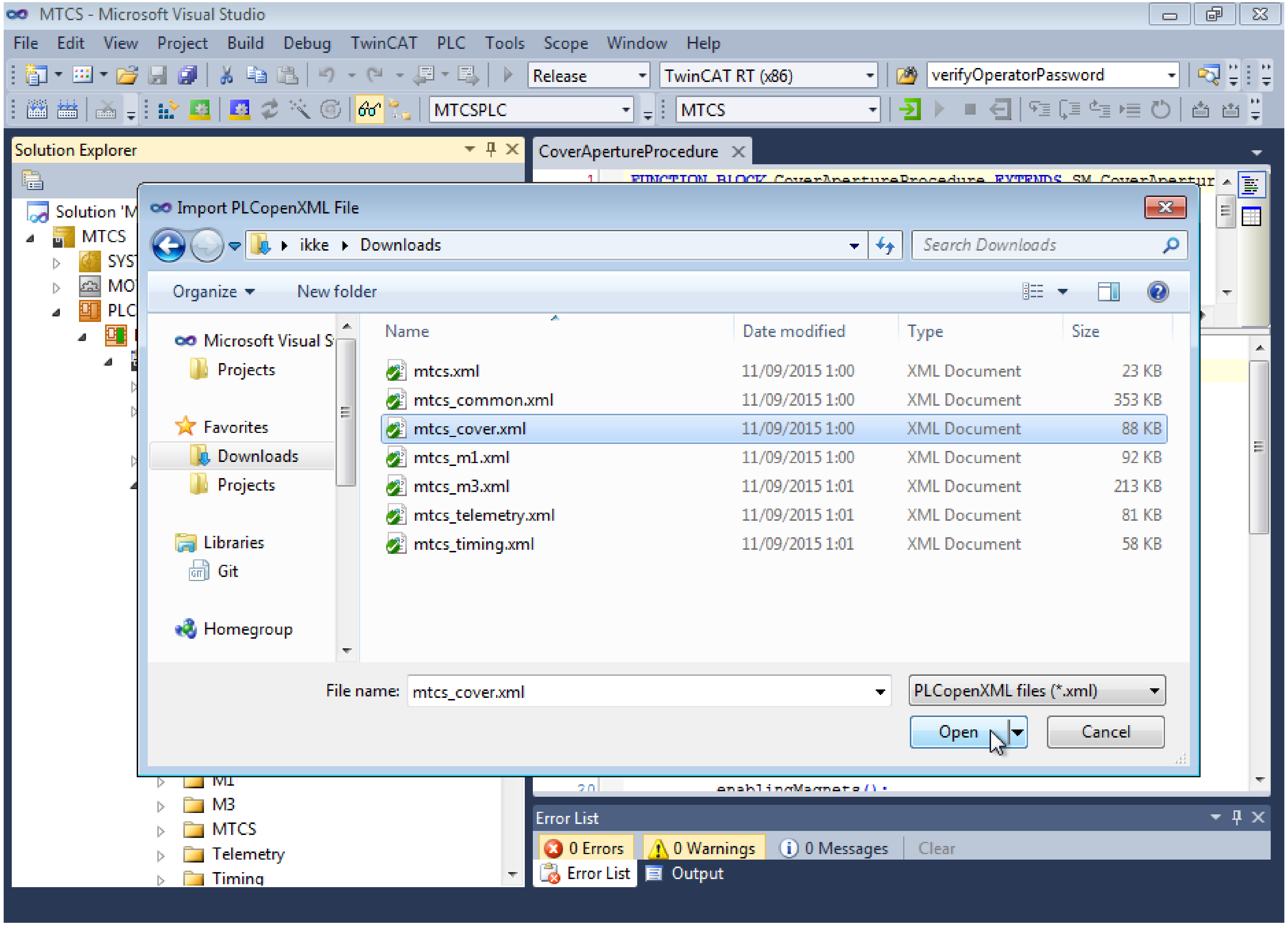
```
_update();

// run the procedure
CASE state OF
    CoverApertureProcedureStates.IDLE:
        _idle();
    CoverApertureProcedureStates.ABORTED:
        _aborted();
    CoverApertureProcedureStates.PREPARE_PROCESS:
        _prepareProcess();
    overApertureProcedureStates.ENABLING_RELAYS:
        _enablingRelays();
    overApertureProcedureStates.ENABLING_MOTORS:
        _enablingMotors();
    overApertureProcedureStates.OPENING_TOP_PANELS:
        _openingTopPanels();
    overApertureProcedureStates.OPENING_BOTH_PANELS:
        _openingBothPanels();
    overApertureProcedureStates.ENABLING_MAGNETS:
        _enablingMagnets();
```

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions



Software design



```
FUNCTION_BLOCK CoverApertureProcedure EXTENDS SM_CoverApertur
    enablingMagnets();
```

Name	Date modified	Type	Size
mtcs.xml	11/09/2015 1:00	XML Document	23 KB
mtcs_common.xml	11/09/2015 1:00	XML Document	353 KB
mtcs_cover.xml	11/09/2015 1:00	XML Document	88 KB
mtcs_m1.xml	11/09/2015 1:00	XML Document	92 KB
mtcs_m3.xml	11/09/2015 1:01	XML Document	213 KB
mtcs_telemetry.xml	11/09/2015 1:01	XML Document	81 KB
mtcs_timing.xml	11/09/2015 1:01	XML Document	58 KB

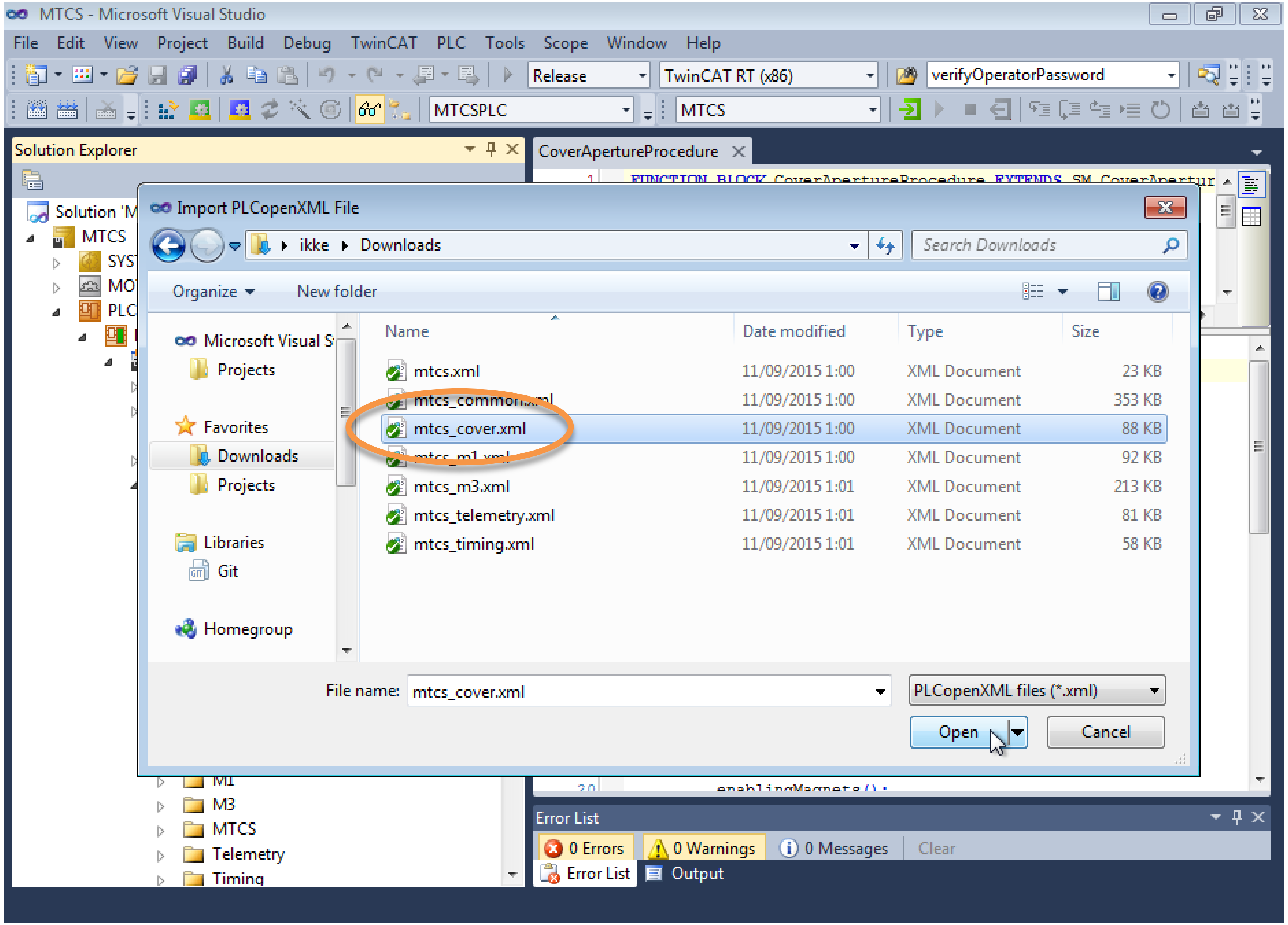
File name: mtcs_cover.xml

PLCopenXML files (*.xml)

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Software design



Name	Date modified	Type	Size
mtcs.xml	11/09/2015 1:00	XML Document	23 KB
mtcs_common.xml	11/09/2015 1:00	XML Document	353 KB
mtcs_cover.xml	11/09/2015 1:00	XML Document	88 KB
mtcs_m1.xml	11/09/2015 1:00	XML Document	92 KB
mtcs_m3.xml	11/09/2015 1:01	XML Document	213 KB
mtcs_telemetry.xml	11/09/2015 1:01	XML Document	81 KB
mtcs_timing.xml	11/09/2015 1:01	XML Document	58 KB

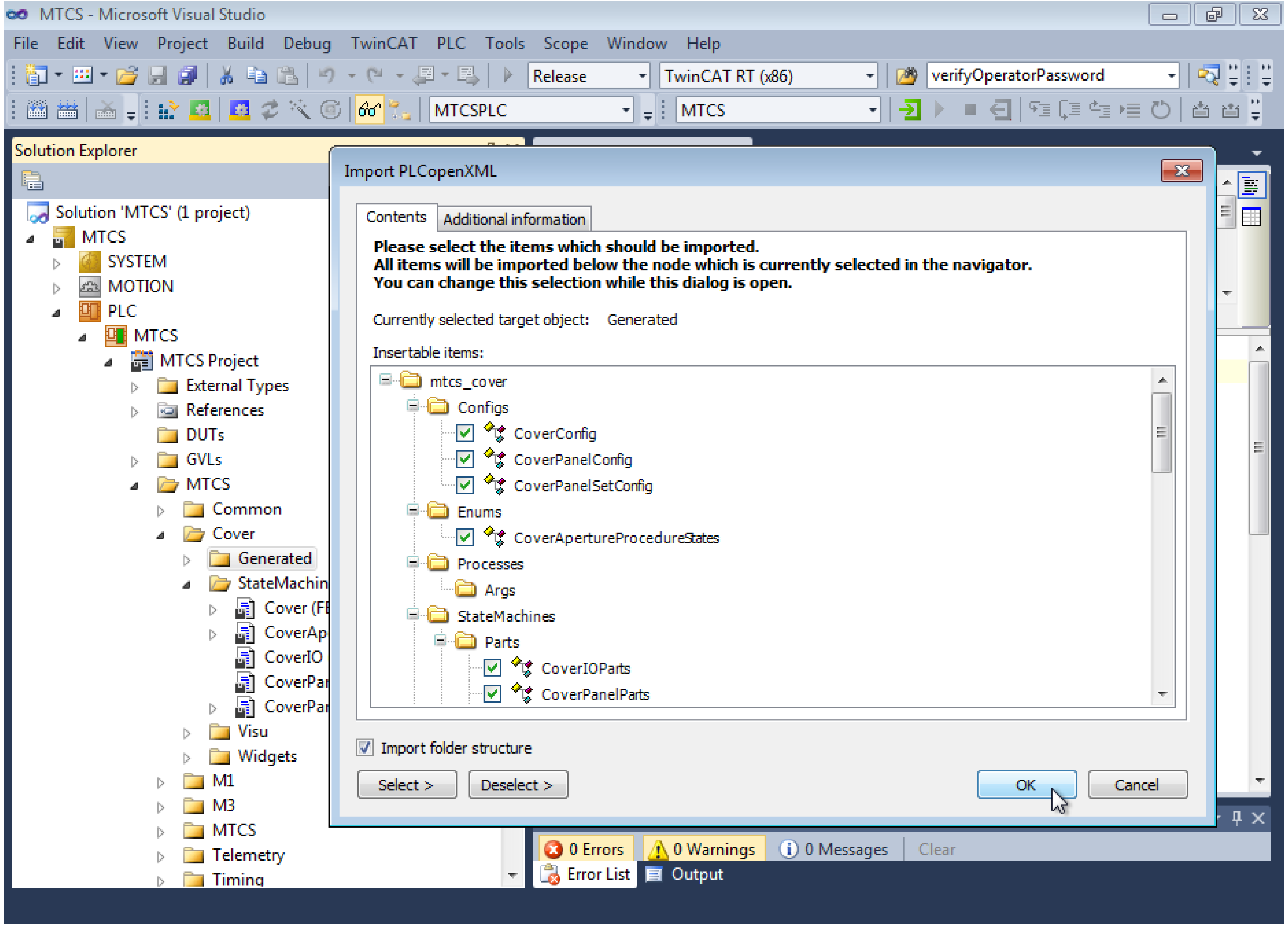
File name: mtcs_cover.xml

PLCopenXML files (*.xml)

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Software design



Import PLCopenXML

Contents | Additional information

Please select the items which should be imported.
All items will be imported below the node which is currently selected in the navigator.
You can change this selection while this dialog is open.

Currently selected target object: Generated

Insertable items:

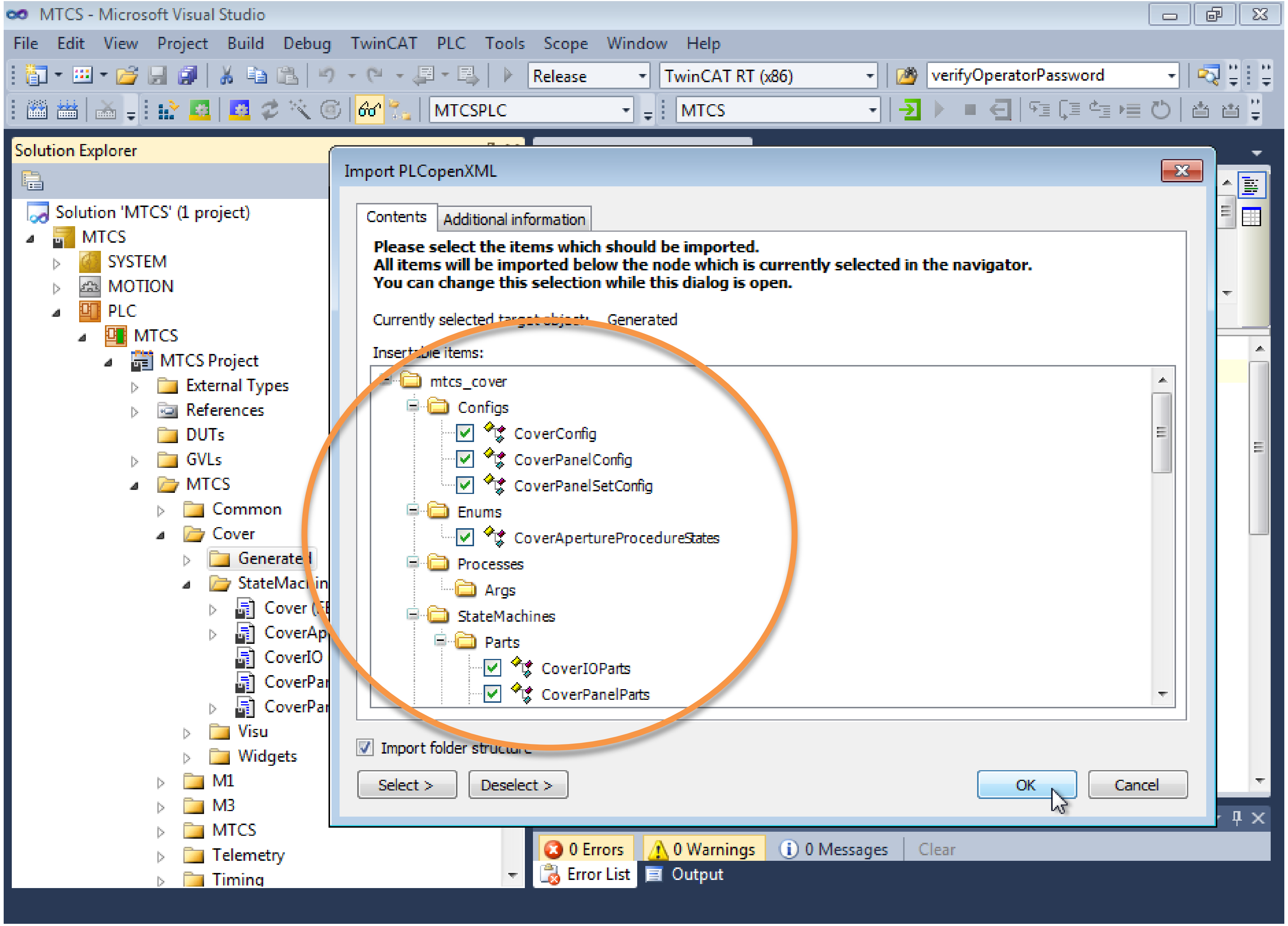
- mtcs_cover
  - Configs
    - ☑ CoverConfig
    - ☑ CoverPanelConfig
    - ☑ CoverPanelSetConfig
  - Enums
    - ☑ CoverApertureProcedureStates
  - Processes
    - Args
  - StateMachines
    - Parts
      - ☑ CoverIOParts
      - ☑ CoverPanelParts

☑ Import folder structure

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Software design



MTCS - Microsoft Visual Studio

Import PLCopenXML

Please select the items which should be imported.
All items will be imported below the node which is currently selected in the navigator.
You can change this selection while this dialog is open.

Currently selected target object: Generated

Insertable items:

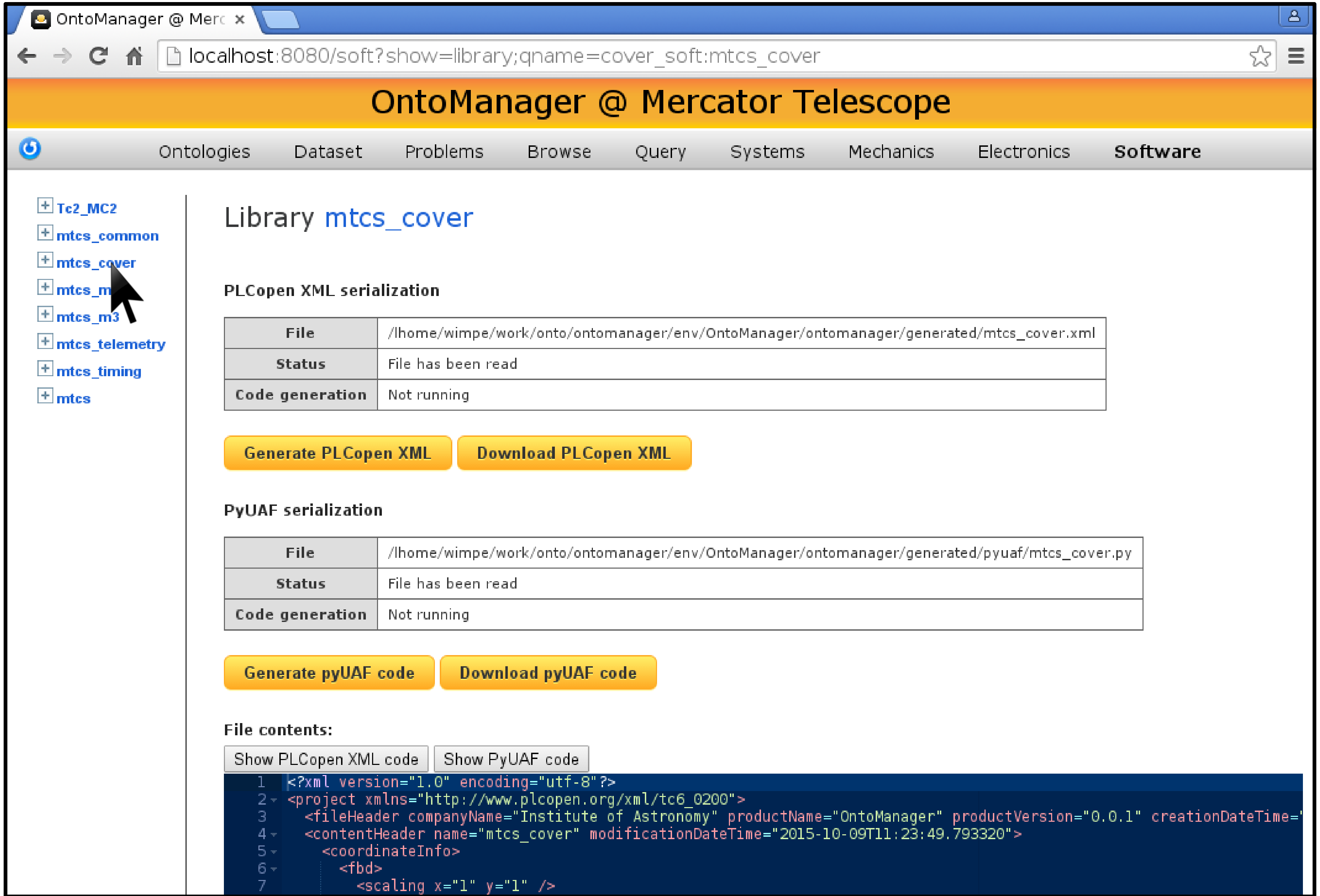
- mtcs_cover
  - Configs
    - ☑ CoverConfig
    - ☑ CoverPanelConfig
    - ☑ CoverPanelSetConfig
  - Enums
    - ☑ CoverApertureProcedureStates
  - Processes
    - Args
  - StateMachines
    - Parts
      - ☑ CoverIOParts
      - ☑ CoverPanelParts

☑ Import folder structure

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Software design



OntoManager @ Mercator Telescope

- Tc2_MC2
- mtcs_common
- mtcs_cover
- mtcs_m
- mtcs_m3
- mtcs_telemetry
- mtcs_timing
- mtcs

Library mtcs_cover

PLCopen XML serialization

File	/lhome/wimpe/work/onto/ontomanager/env/OntoManager/ontomanager/generated/mtcs_cover.xml
Status	File has been read
Code generation	Not running

PyUAF serialization

File	/lhome/wimpe/work/onto/ontomanager/env/OntoManager/ontomanager/generated/pyuaf/mtcs_cover.py
Status	File has been read
Code generation	Not running

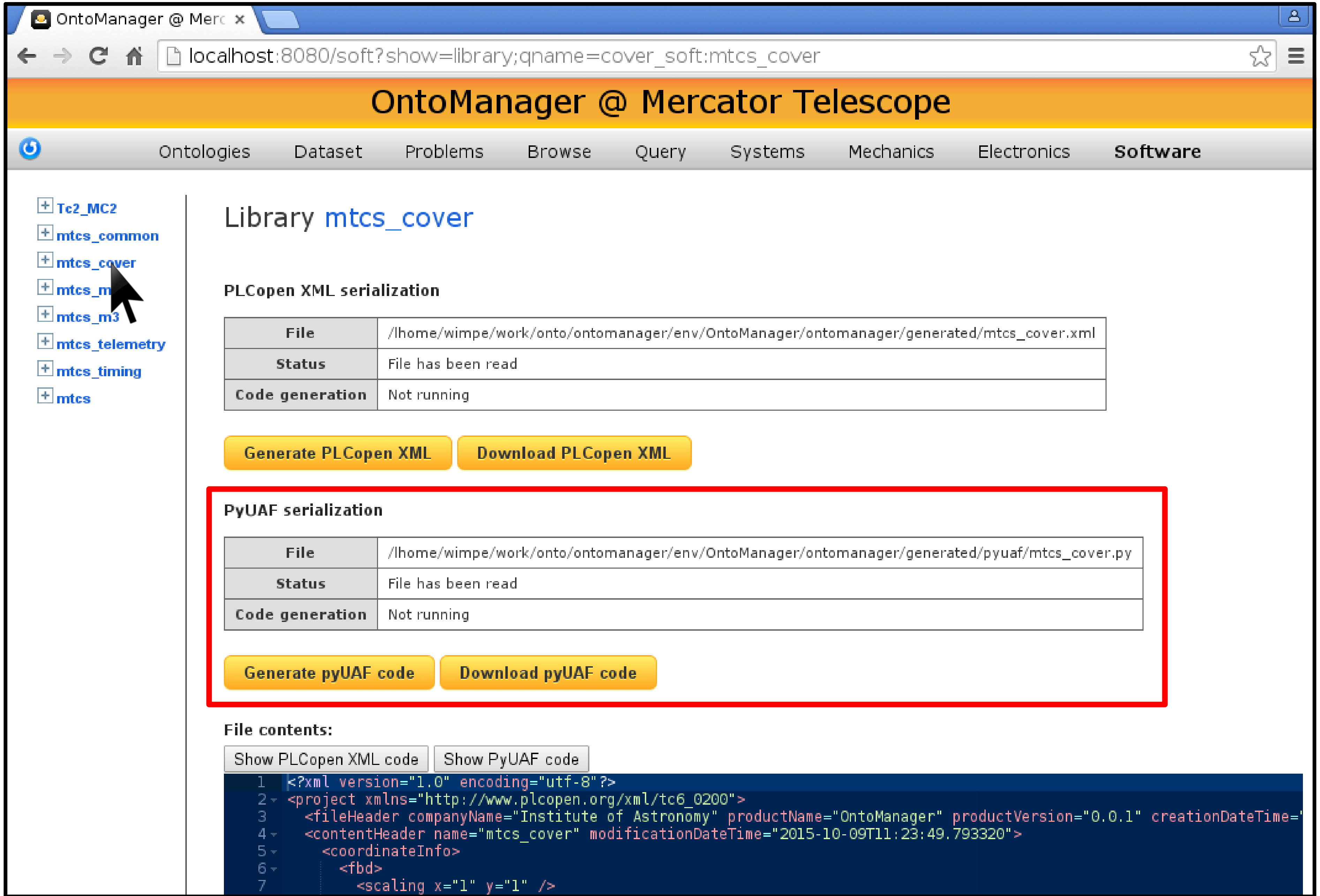
File contents:

```
<?xml version="1.0" encoding="utf-8"?>
<project xmlns="http://www.plcopen.org/xml/tc6_0200">
  <fileHeader companyName="Institute of Astronomy" productName="OntoManager" productVersion="0.0.1" creationDateTime="
  <contentHeader name="mtcs_cover" modificationDateTime="2015-10-09T11:23:49.793320">
    <coordinateInfo>
      <fbd>
        <scaling x="1" y="1" />
```

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Software design



OntoManager @ Merc ×
localhost:8080/soft?show=library;qname=cover_soft:mtcs_cover

OntoManager @ Mercator Telescope

Library mtcs_cover

PLCopen XML serialization

File	/lhome/wimpe/work/onto/ontomanager/env/OntoManager/ontomanager/generated/mtcs_cover.xml
Status	File has been read
Code generation	Not running

PyUAF serialization

File	/lhome/wimpe/work/onto/ontomanager/env/OntoManager/ontomanager/generated/pyuaf/mtcs_cover.py
Status	File has been read
Code generation	Not running

File contents:

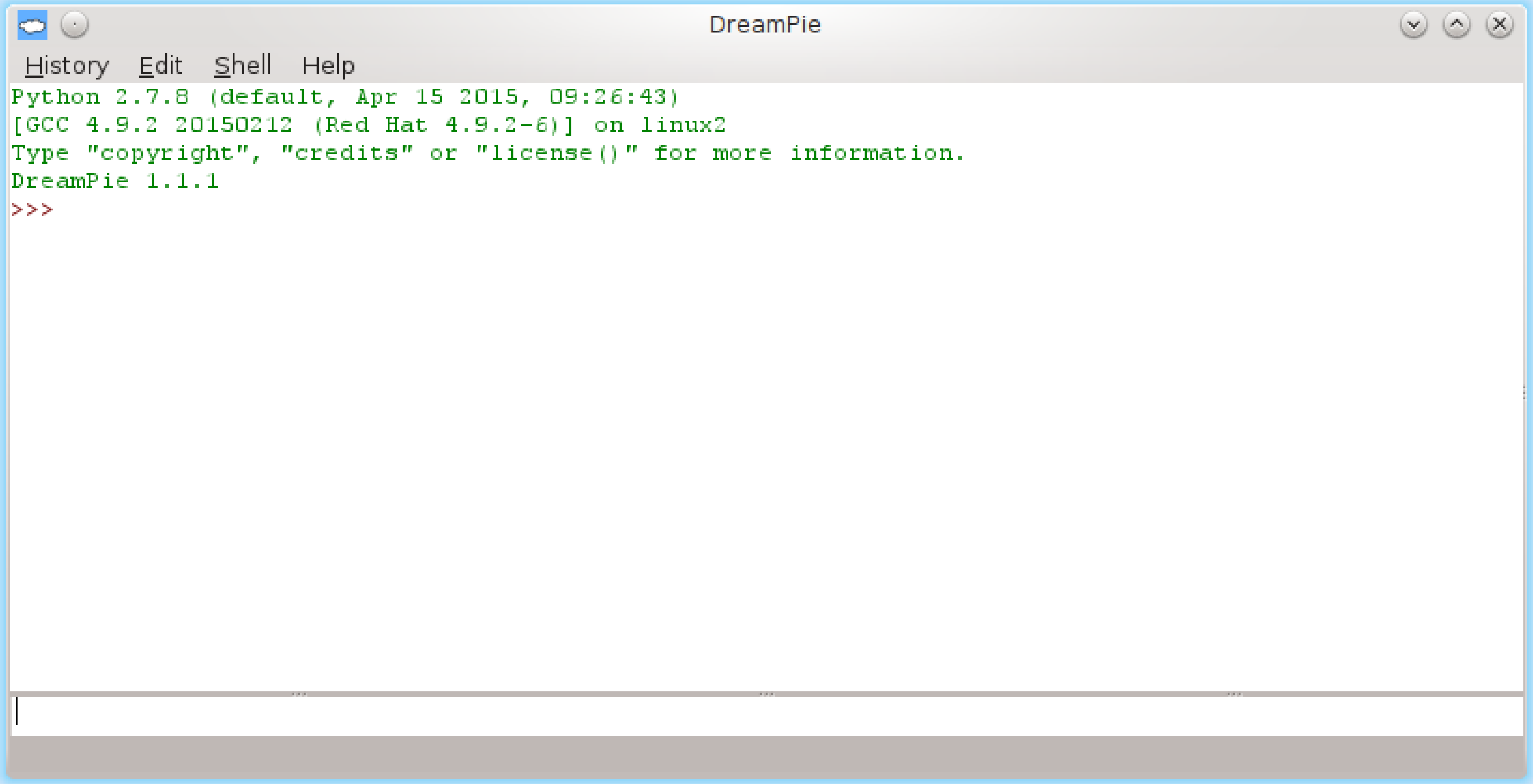
```
<?xml version="1.0" encoding="utf-8"?>
<project xmlns="http://www.plcopen.org/xml/tc6_0200">
  <fileHeader companyName="Institute of Astronomy" productName="OntoManager" productVersion="0.0.1" creationDateTime="
  <contentHeader name="mtcs_cover" modificationDateTime="2015-10-09T11:23:49.793320">
    <coordinateInfo>
      <fbd>
        <scaling x="1" y="1" />
```

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Software design

- Generated Python code (client side)
  - Based on our OPC UA library “UAF”: http://github.com/uaf/uaf



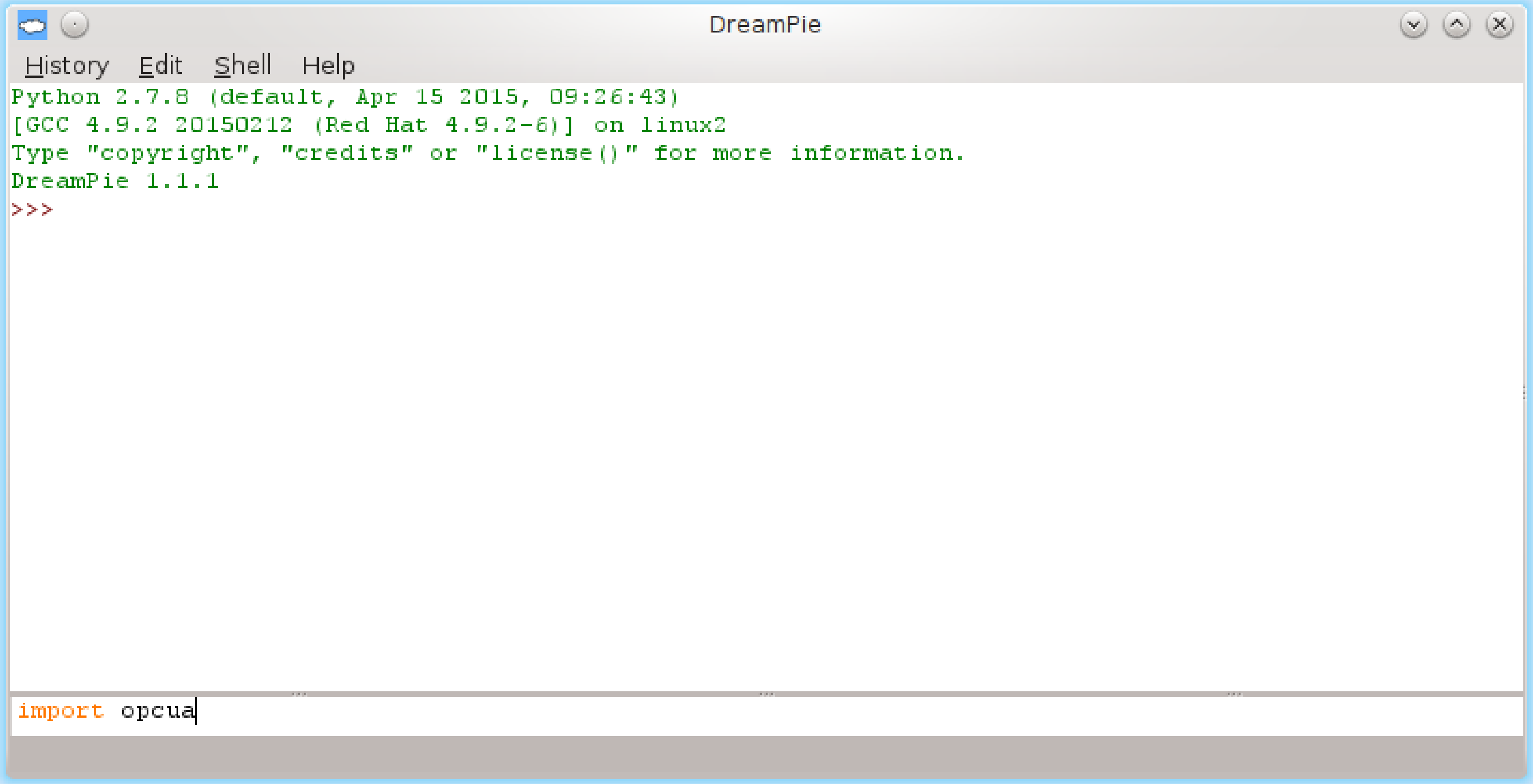
```
Python 2.7.8 (default, Apr 15 2015, 09:26:43)
[GCC 4.9.2 20150212 (Red Hat 4.9.2-6)] on linux2
Type "copyright", "credits" or "license()" for more information.
DreamPie 1.1.1
>>>
```

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Software design

- Generated Python code (client side)
  - Based on our OPC UA library "UAF": http://github.com/uaf/uaf

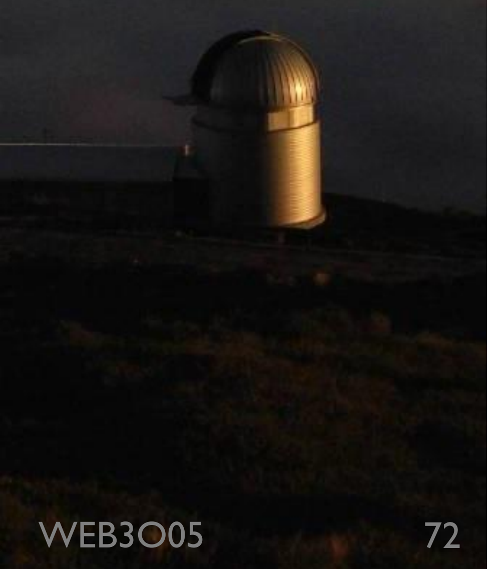


```
Python 2.7.8 (default, Apr 15 2015, 09:26:43)
[GCC 4.9.2 20150212 (Red Hat 4.9.2-6)] on linux2
Type "copyright", "credits" or "license()" for more information.
DreamPie 1.1.1
>>>
```

```
import opcua
```

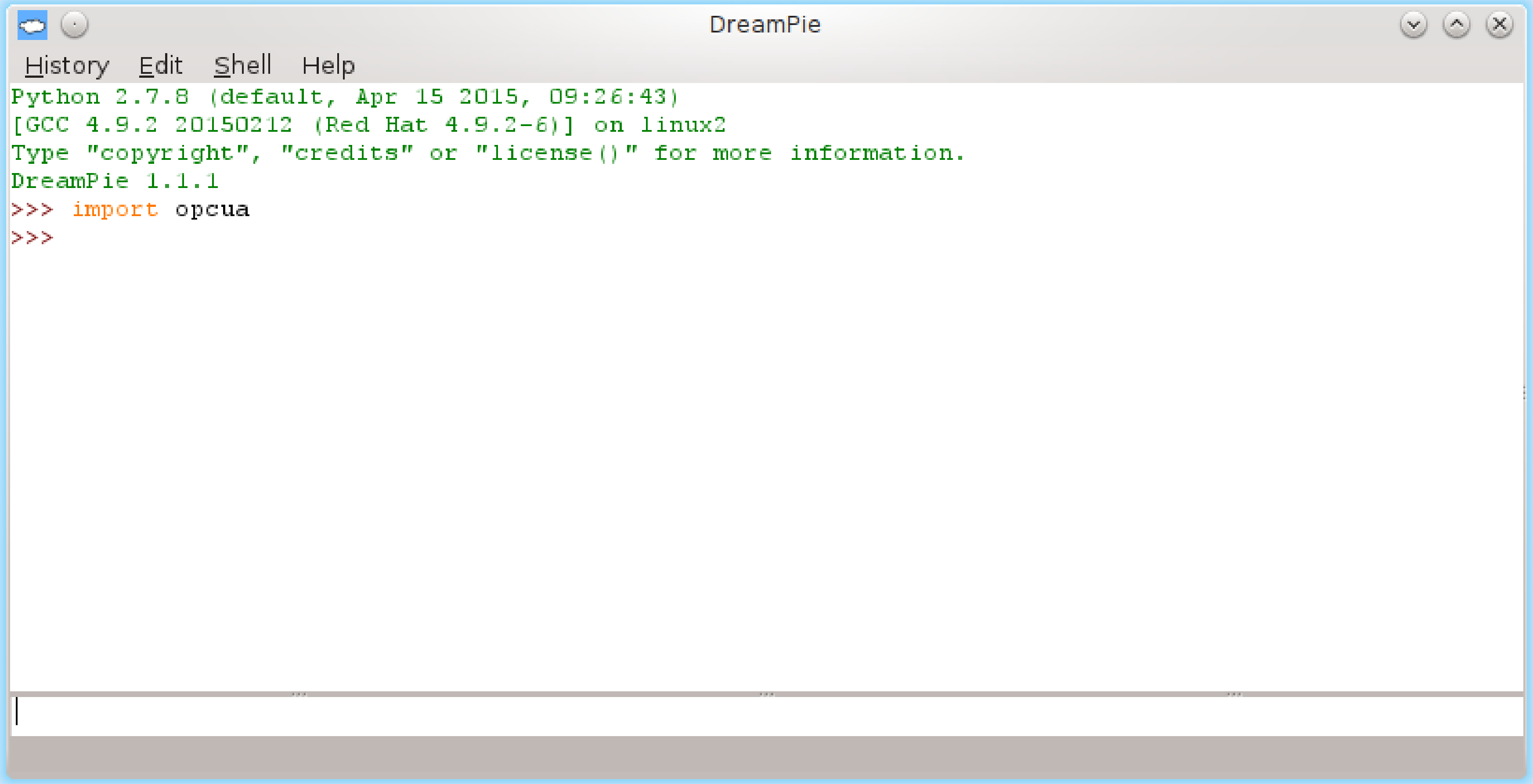
Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions



Software design

- Generated Python code (client side)
  - Based on our OPC UA library "UAF": http://github.com/uaf/uaf



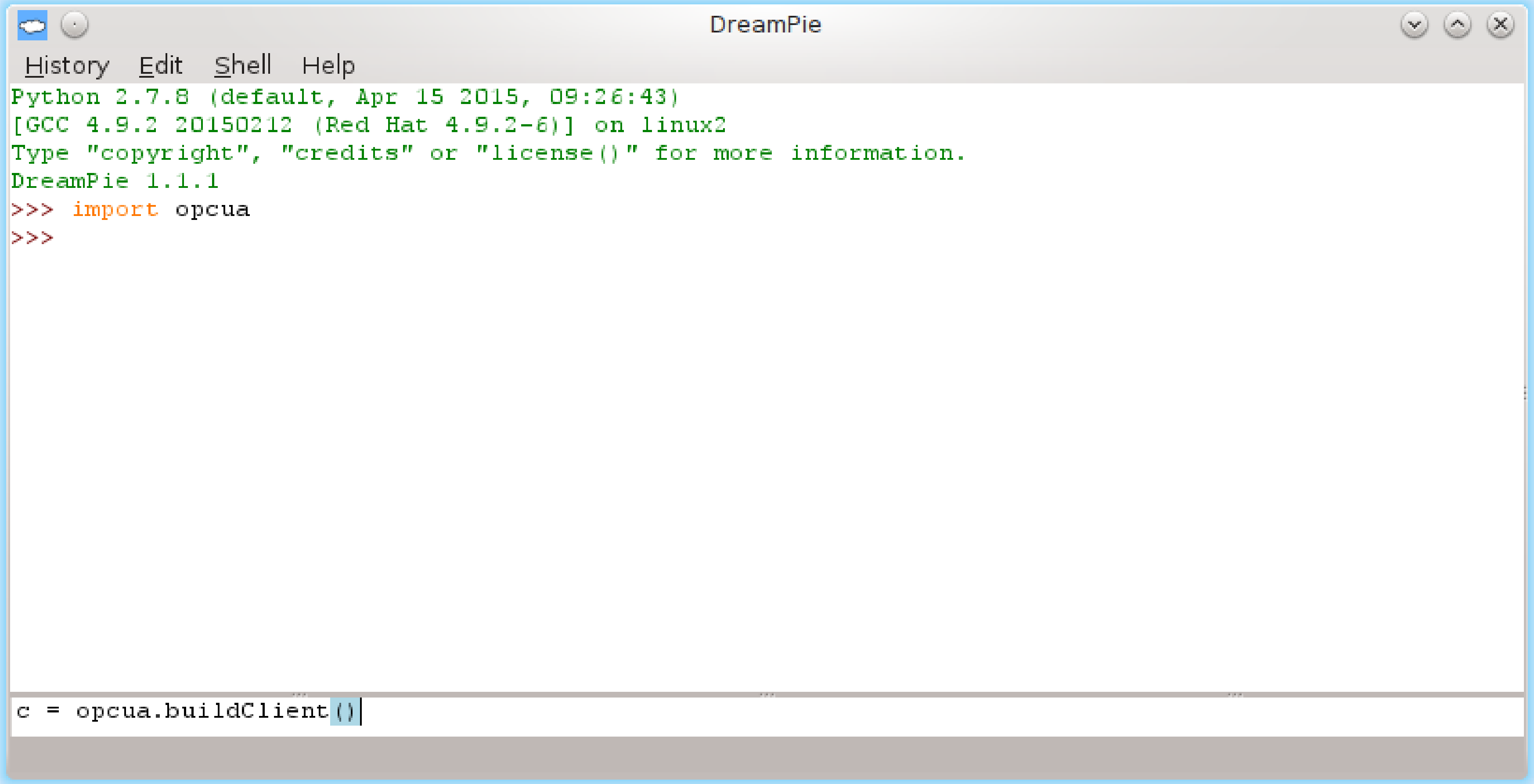
```
Python 2.7.8 (default, Apr 15 2015, 09:26:43)
[GCC 4.9.2 20150212 (Red Hat 4.9.2-6)] on linux2
Type "copyright", "credits" or "license()" for more information.
DreamPie 1.1.1
>>> import opcua
>>>
```

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

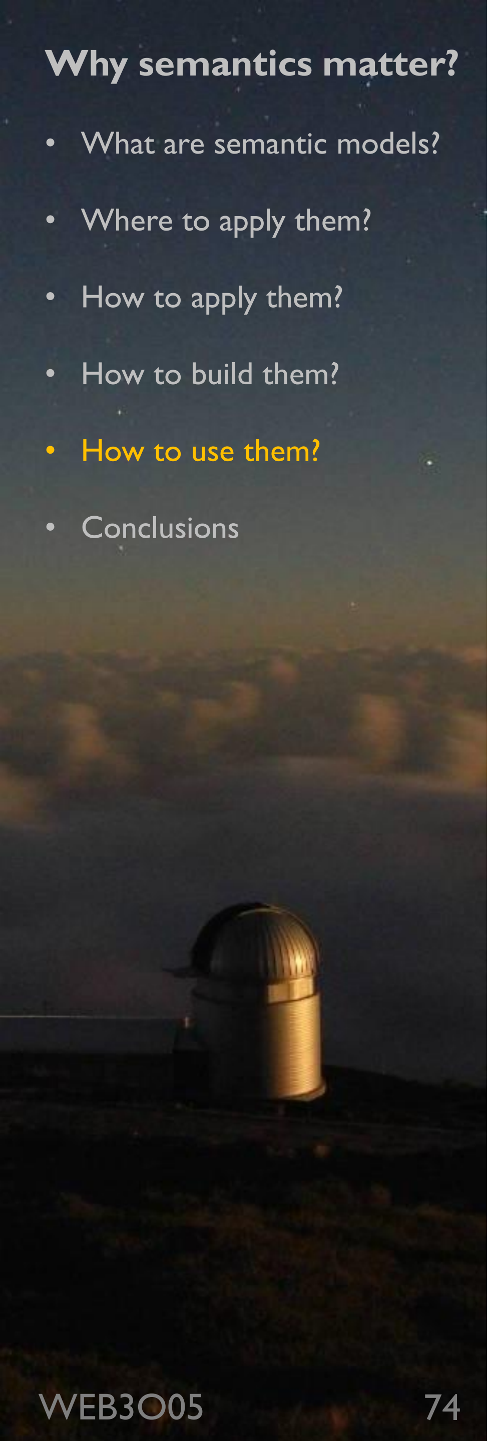
Software design

- Generated Python code (client side)
  - Based on our OPC UA library “UAF”: http://github.com/uaf/uaf



```
Python 2.7.8 (default, Apr 15 2015, 09:26:43)
[GCC 4.9.2 20150212 (Red Hat 4.9.2-6)] on linux2
Type "copyright", "credits" or "license()" for more information.
DreamPie 1.1.1
>>> import opcua
>>>
```

```
c = opcua.buildClient()
```

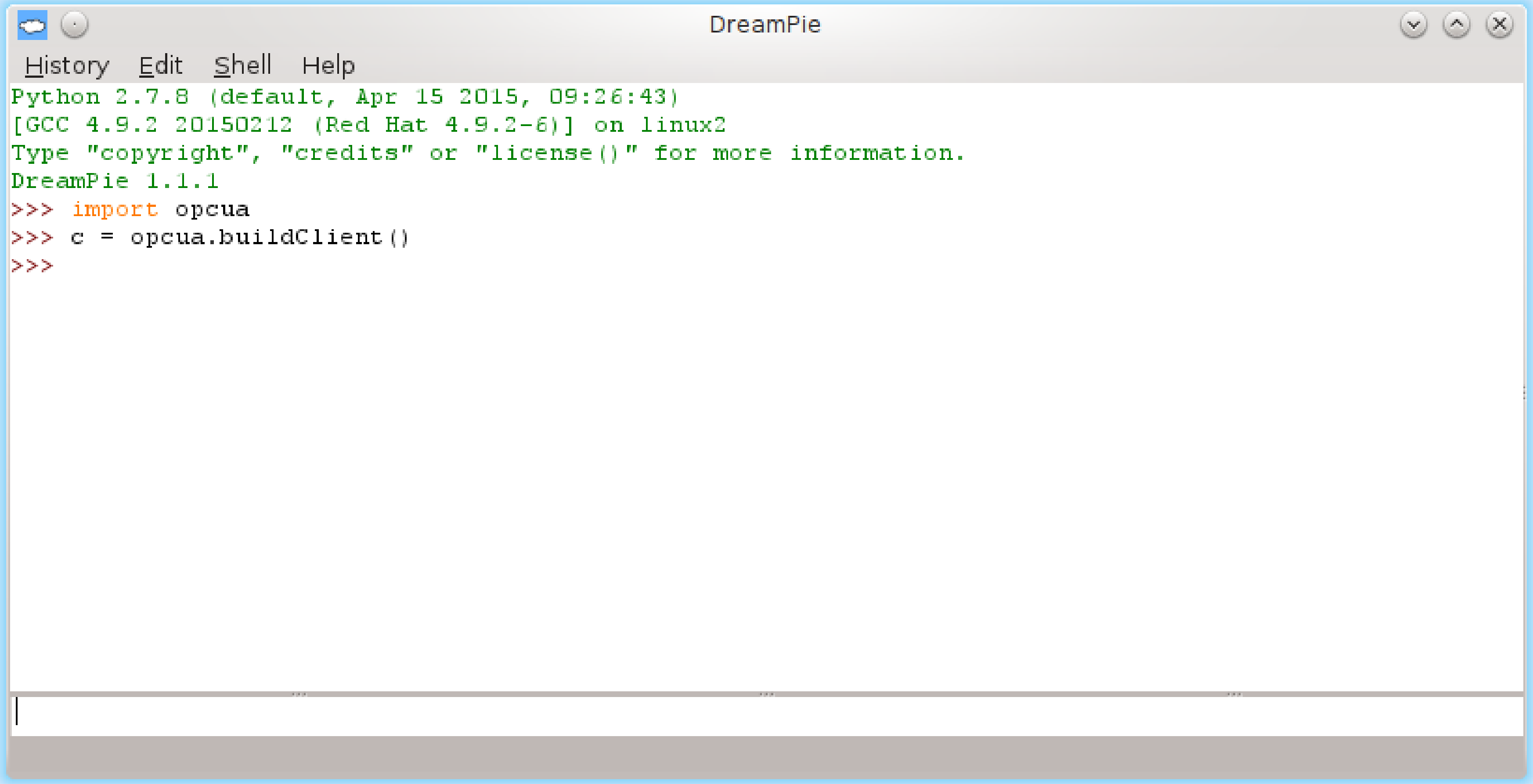


Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Software design

- Generated Python code (client side)
  - Based on our OPC UA library “UAF”: http://github.com/uaf/uaf



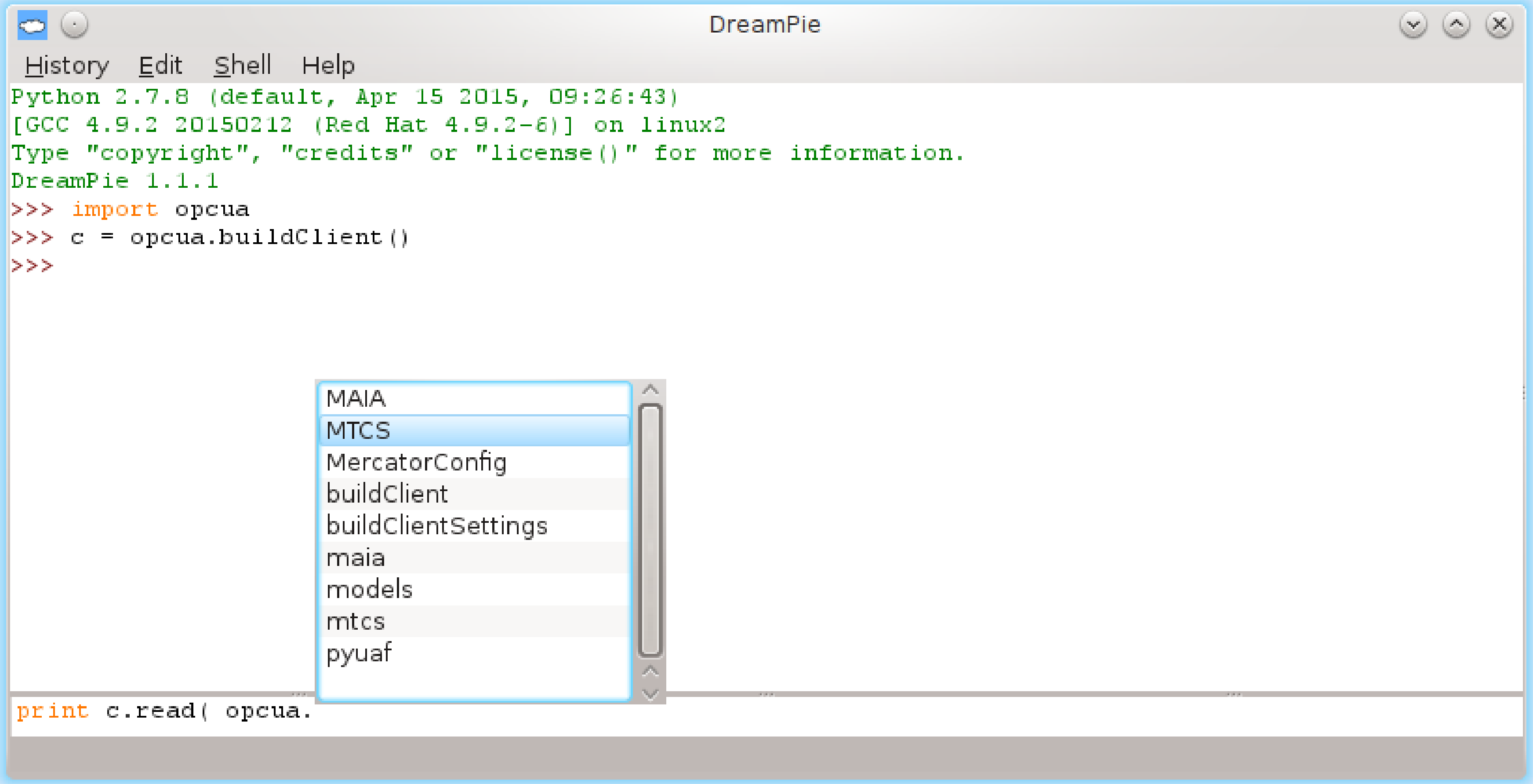
```
Python 2.7.8 (default, Apr 15 2015, 09:26:43)
[GCC 4.9.2 20150212 (Red Hat 4.9.2-6)] on linux2
Type "copyright", "credits" or "license()" for more information.
DreamPie 1.1.1
>>> import opcua
>>> c = opcua.buildClient()
>>>
```

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Software design

- Generated Python code (client side)
  - Based on our OPC UA library "UAF": http://github.com/uaf/uaf



```
Python 2.7.8 (default, Apr 15 2015, 09:26:43)
[GCC 4.9.2 20150212 (Red Hat 4.9.2-6)] on linux2
Type "copyright", "credits" or "license()" for more information.
DreamPie 1.1.1
>>> import opcua
>>> c = opcua.buildClient()
>>>
```

```
MAIA
MTCS
MercatorConfig
buildClient
buildClientSettings
maia
models
mtcs
pyuaf
```

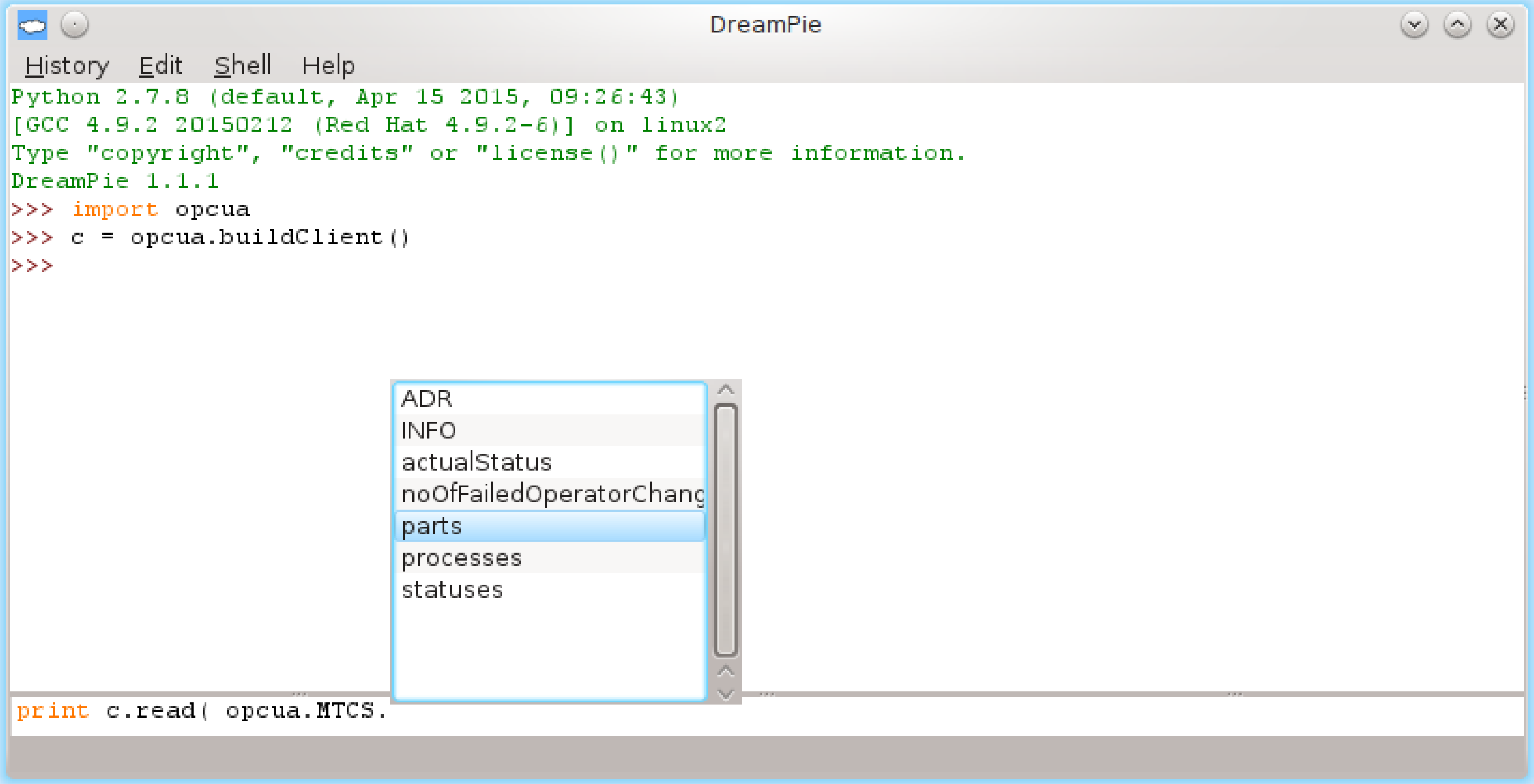
```
print c.read( opcua.
```

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Software design

- Generated Python code (client side)
  - Based on our OPC UA library “UAF”: http://github.com/uaf/uaf



```
Python 2.7.8 (default, Apr 15 2015, 09:26:43)
[GCC 4.9.2 20150212 (Red Hat 4.9.2-6)] on linux2
Type "copyright", "credits" or "license()" for more information.
DreamPie 1.1.1
>>> import opcua
>>> c = opcua.buildClient()
>>>
```

```
ADR
INFO
actualStatus
noOfFailedOperatorChang
parts
processes
statuses
```

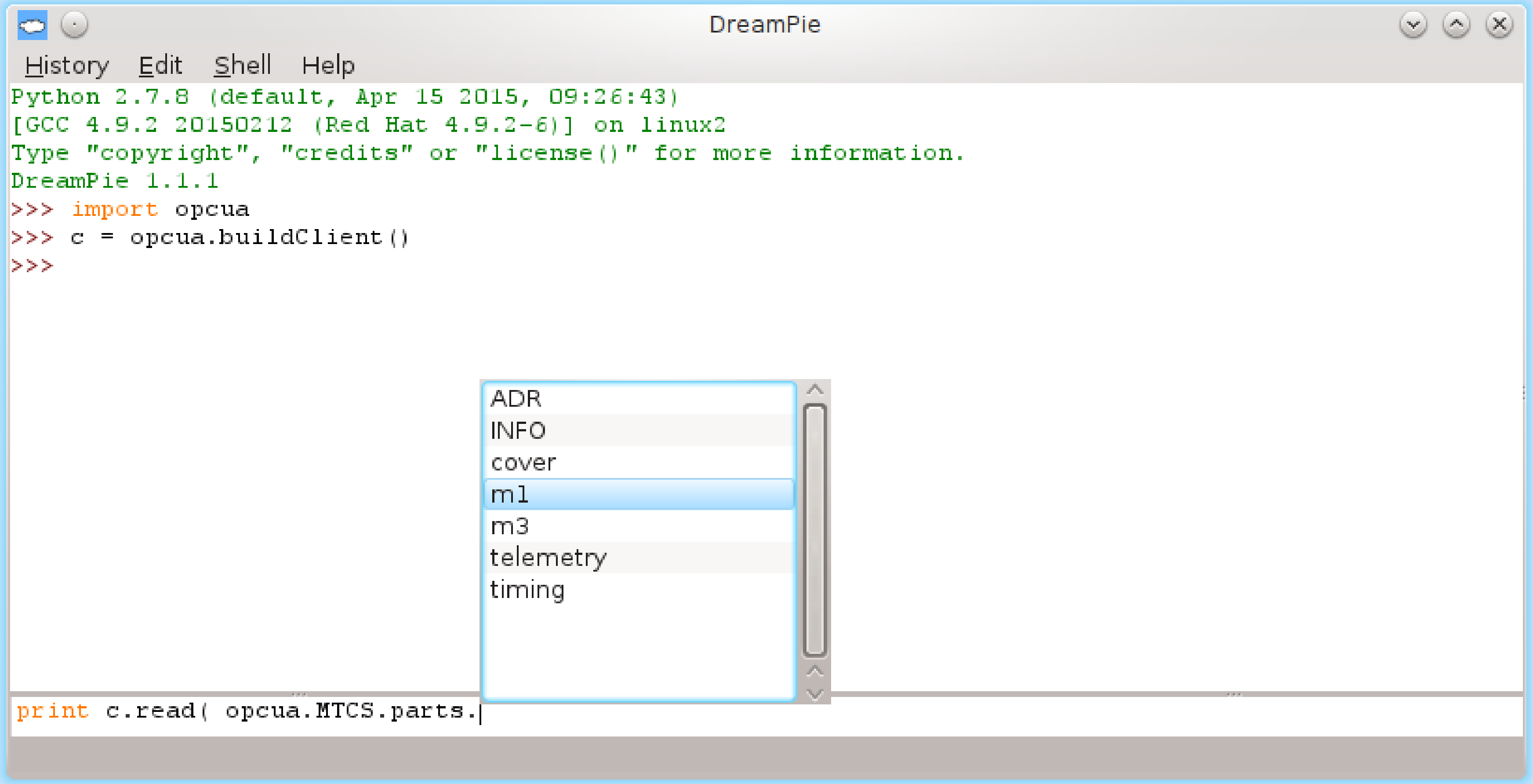
```
print c.read( opcua.MTCS.
```

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Software design

- Generated Python code (client side)
  - Based on our OPC UA library "UAF": http://github.com/uaf/uaf



```
Python 2.7.8 (default, Apr 15 2015, 09:26:43)
[GCC 4.9.2 20150212 (Red Hat 4.9.2-6)] on linux2
Type "copyright", "credits" or "license()" for more information.
DreamPie 1.1.1
>>> import opcua
>>> c = opcua.buildClient()
>>>
```

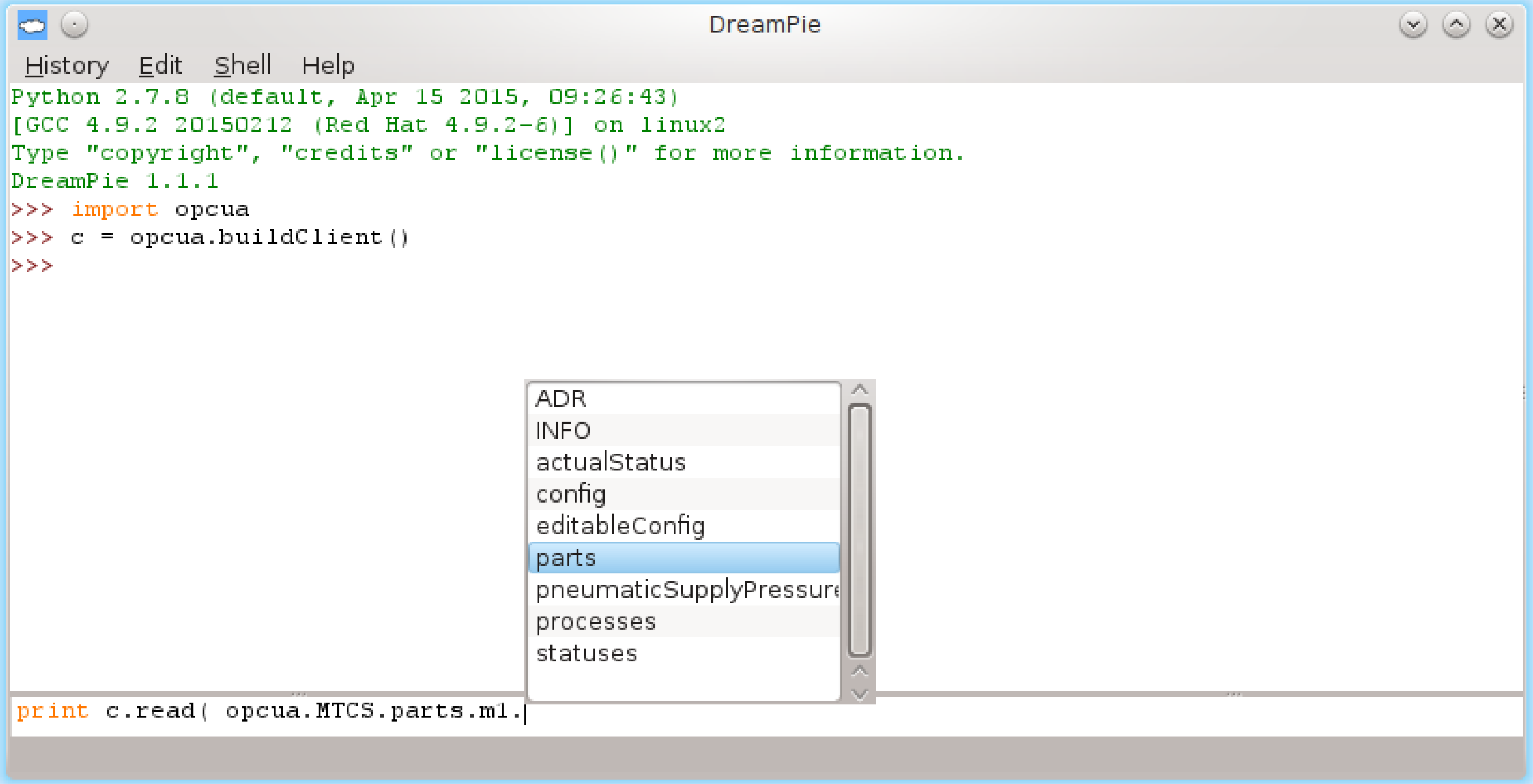
```
print c.read( opcua.MTCS.parts.
```

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Software design

- Generated Python code (client side)
  - Based on our OPC UA library “UAF”: http://github.com/uaf/uaf



```
Python 2.7.8 (default, Apr 15 2015, 09:26:43)
[GCC 4.9.2 20150212 (Red Hat 4.9.2-6)] on linux2
Type "copyright", "credits" or "license()" for more information.
DreamPie 1.1.1
>>> import opcua
>>> c = opcua.buildClient()
>>>
```

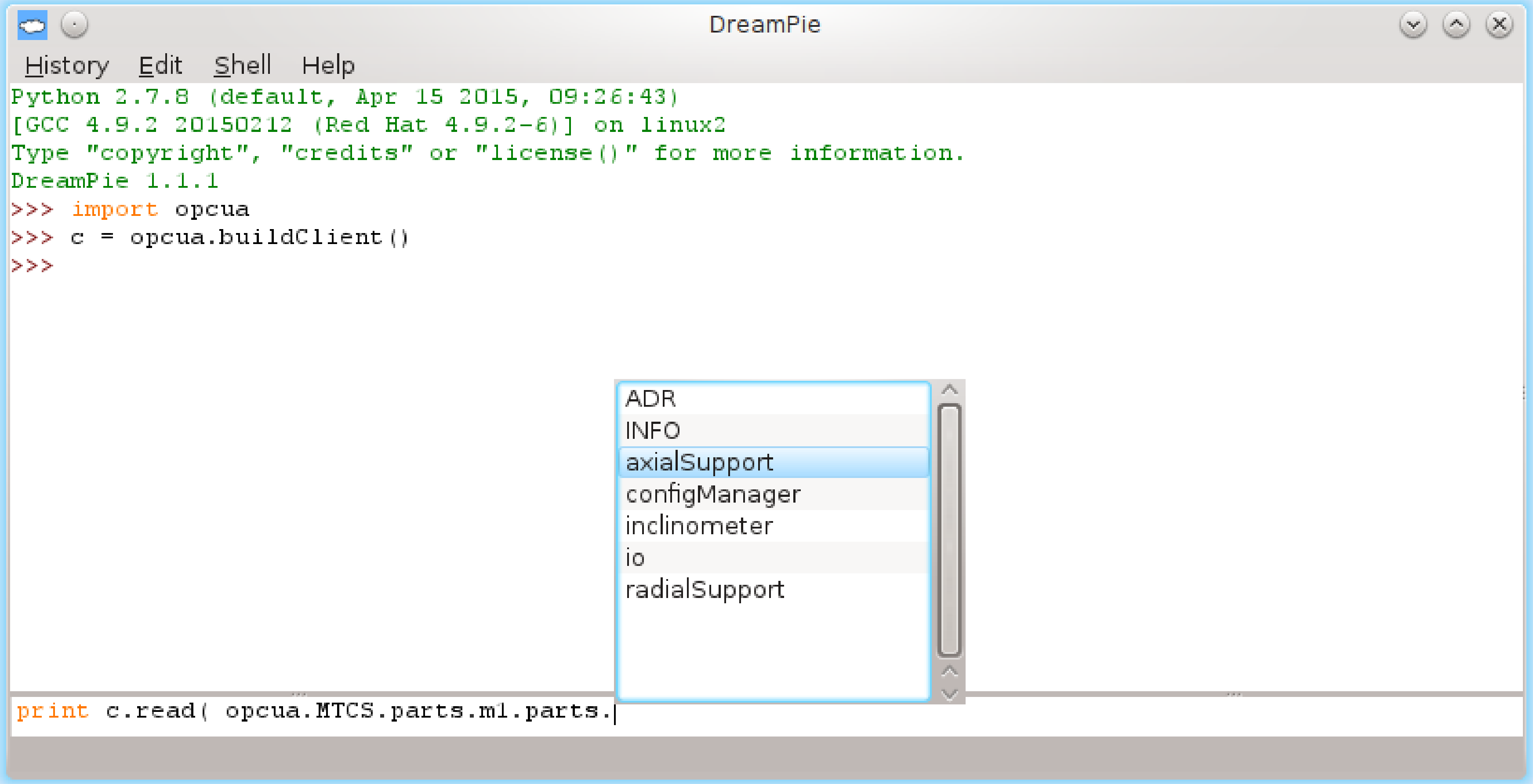
```
print c.read( opcua.MTCS.parts.m1.
```

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Software design

- Generated Python code (client side)
  - Based on our OPC UA library “UAF”: http://github.com/uaf/uaf



```
Python 2.7.8 (default, Apr 15 2015, 09:26:43)
[GCC 4.9.2 20150212 (Red Hat 4.9.2-6)] on linux2
Type "copyright", "credits" or "license()" for more information.
DreamPie 1.1.1
>>> import opcua
>>> c = opcua.buildClient()
>>>
```

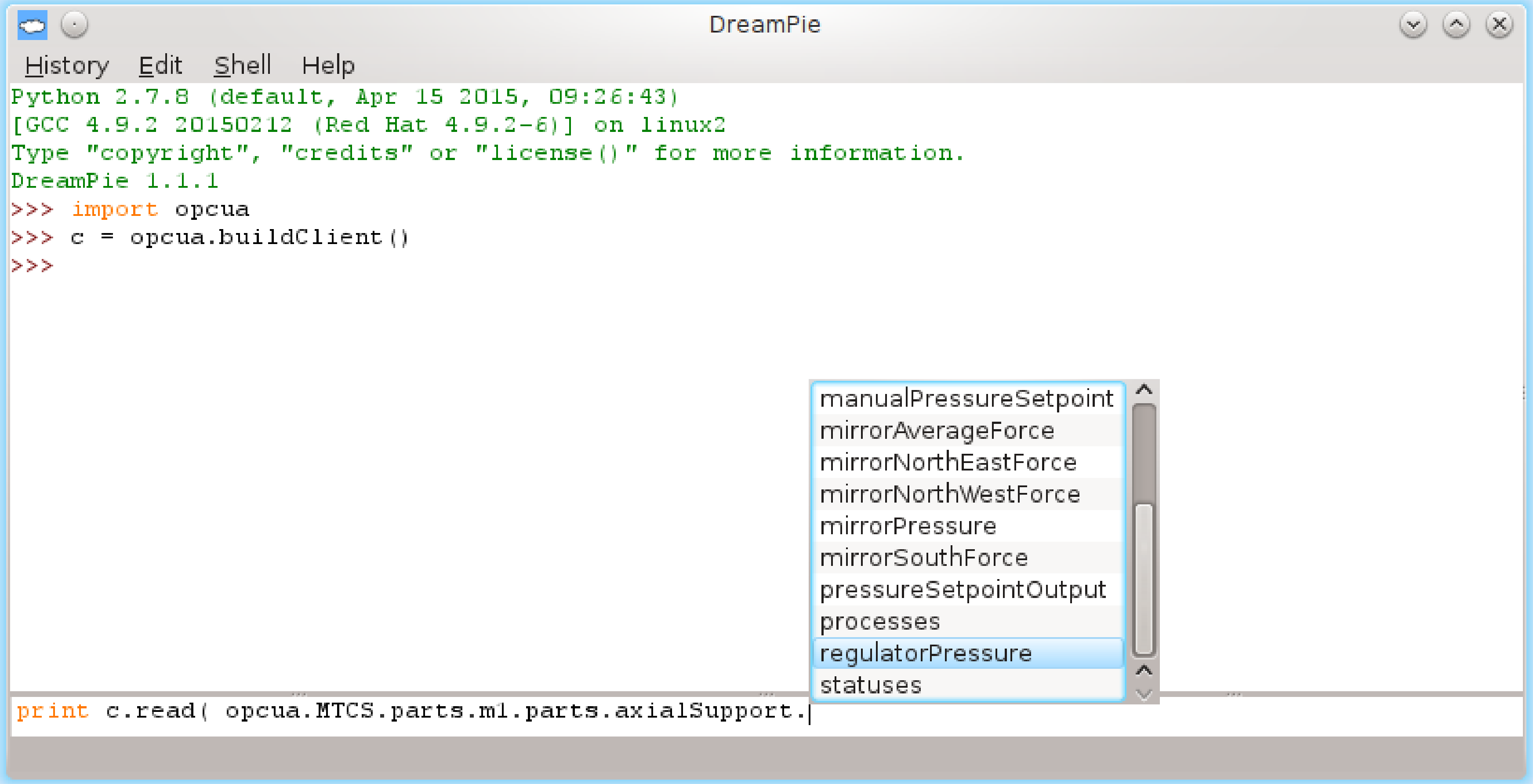
```
print c.read( opcua.MTCS.parts.m1.parts.
```

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Software design

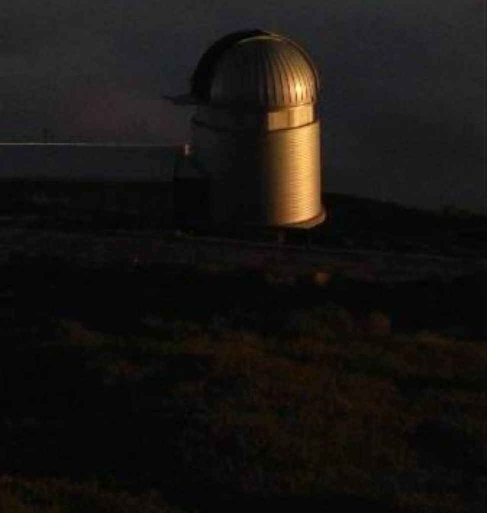
- Generated Python code (client side)
  - Based on our OPC UA library “UAF”: http://github.com/uaf/uaf

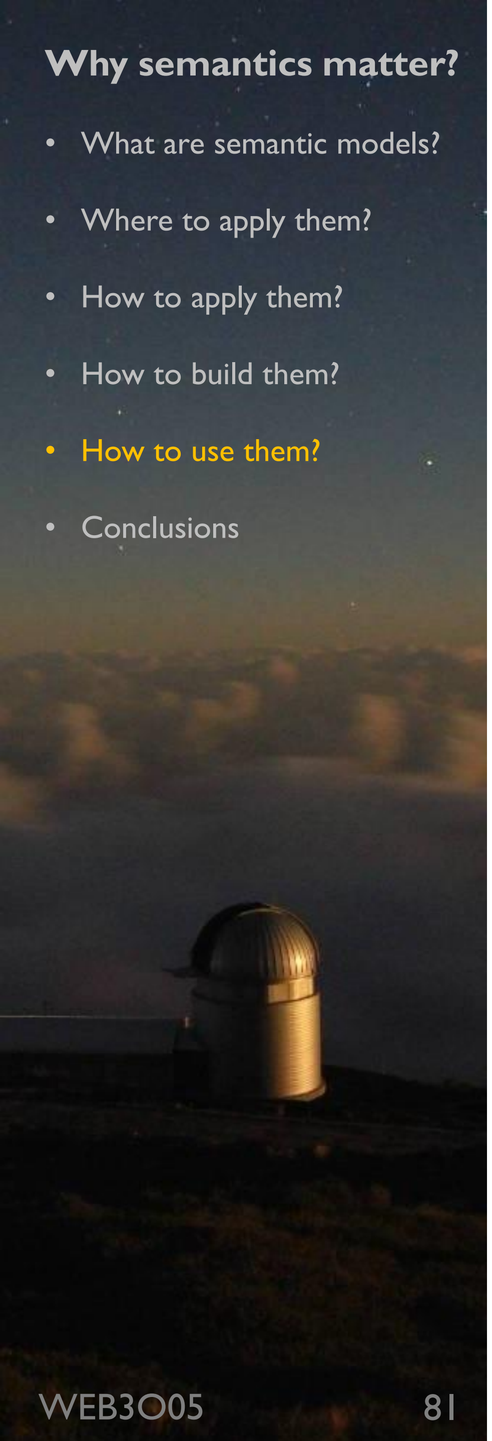


```
Python 2.7.8 (default, Apr 15 2015, 09:26:43)
[GCC 4.9.2 20150212 (Red Hat 4.9.2-6)] on linux2
Type "copyright", "credits" or "license()" for more information.
DreamPie 1.1.1
>>> import opcua
>>> c = opcua.buildClient()
>>>
```

```
manualPressureSetpoint
mirrorAverageForce
mirrorNorthEastForce
mirrorNorthWestForce
mirrorPressure
mirrorSouthForce
pressureSetpointOutput
processes
regulatorPressure
statuses
```

```
print c.read( opcua.MTCS.parts.m1.parts.axialSupport.
```



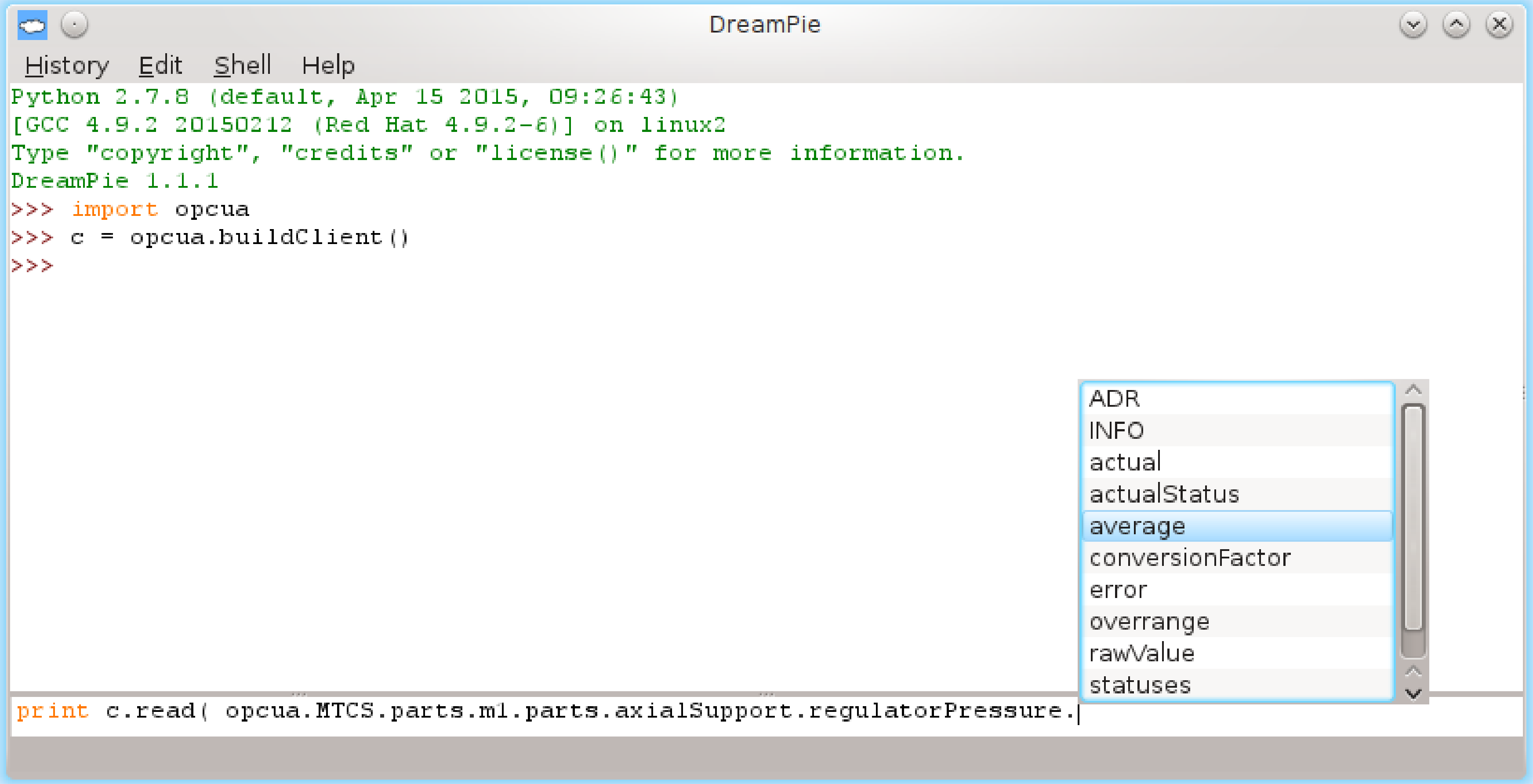


Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Software design

- Generated Python code (client side)
  - Based on our OPC UA library "UAF": http://github.com/uaf/uaf



```
Python 2.7.8 (default, Apr 15 2015, 09:26:43)
[GCC 4.9.2 20150212 (Red Hat 4.9.2-6)] on linux2
Type "copyright", "credits" or "license()" for more information.
DreamPie 1.1.1
>>> import opcua
>>> c = opcua.buildClient()
>>>
```

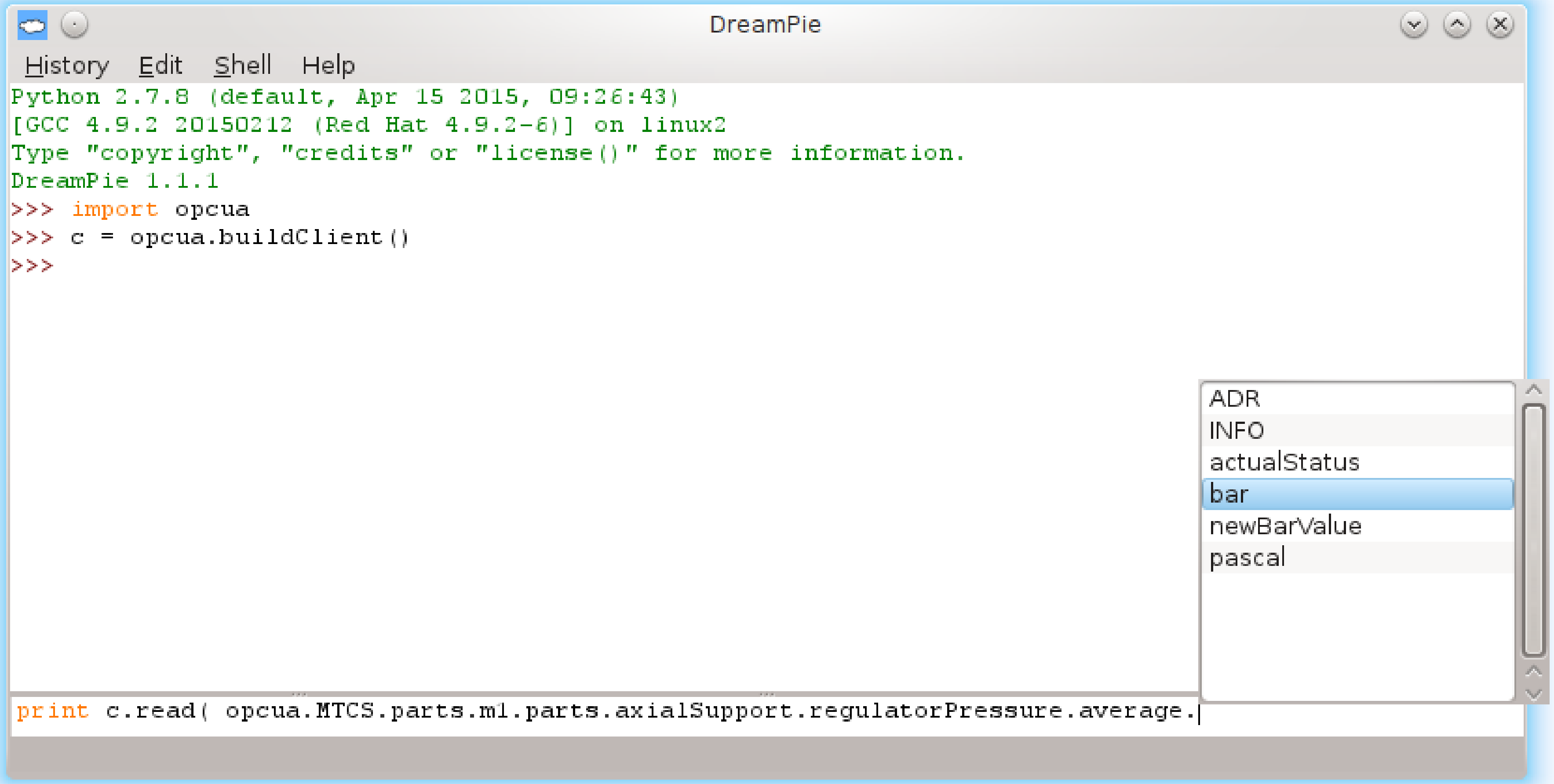
```
print c.read( opcua.MTCS.parts.m1.parts.axialSupport.regulatorPressure.
```

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

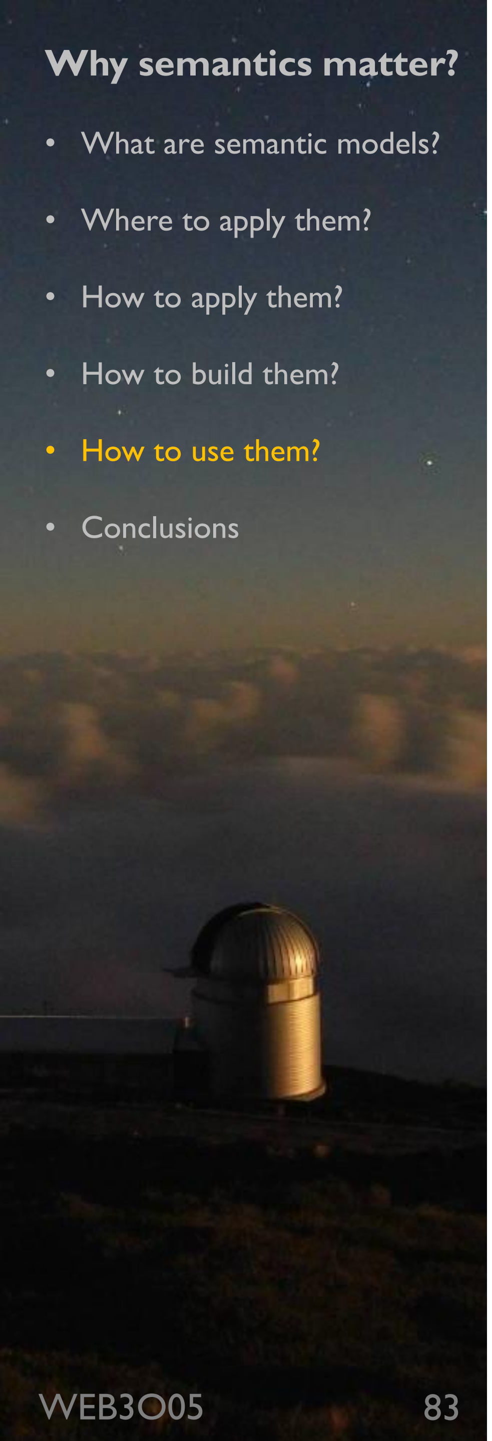
Software design

- Generated Python code (client side)
  - Based on our OPC UA library "UAF": http://github.com/uaf/uaf



```
Python 2.7.8 (default, Apr 15 2015, 09:26:43)
[GCC 4.9.2 20150212 (Red Hat 4.9.2-6)] on linux2
Type "copyright", "credits" or "license()" for more information.
DreamPie 1.1.1
>>> import opcua
>>> c = opcua.buildClient()
>>>
```

```
print c.read( opcua.MTCS.parts.m1.parts.axialSupport.regulatorPressure.average.
```

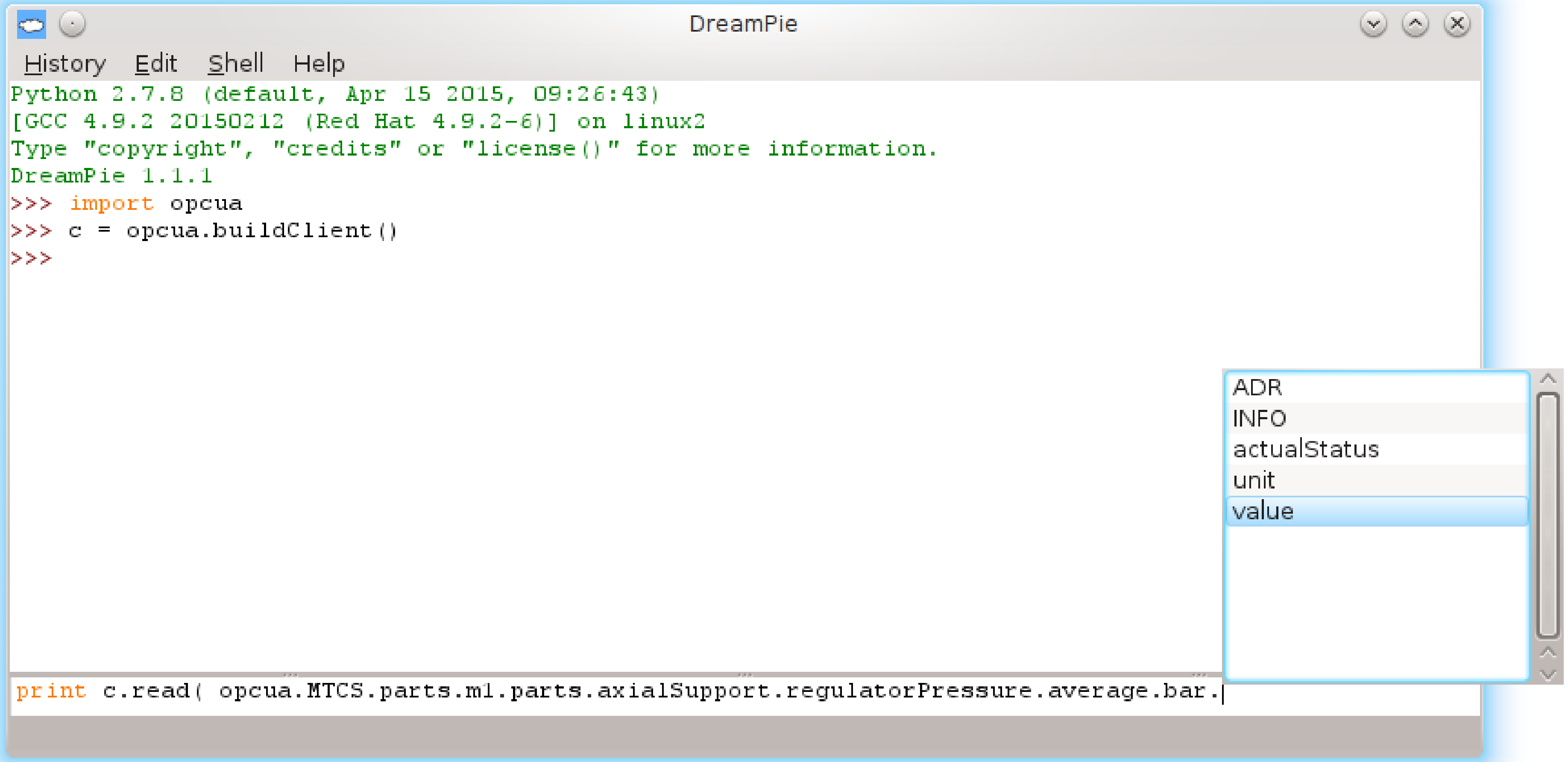


Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Software design

- Generated Python code (client side)
  - Based on our OPC UA library "UAF": http://github.com/uaf/uaf



```
Python 2.7.8 (default, Apr 15 2015, 09:26:43)
[GCC 4.9.2 20150212 (Red Hat 4.9.2-6)] on linux2
Type "copyright", "credits" or "license()" for more information.
DreamPie 1.1.1
>>> import opcua
>>> c = opcua.buildClient()
>>>
```

```
print c.read( opcua.MTCS.parts.m1.parts.axialSupport.regulatorPressure.average.bar.
```

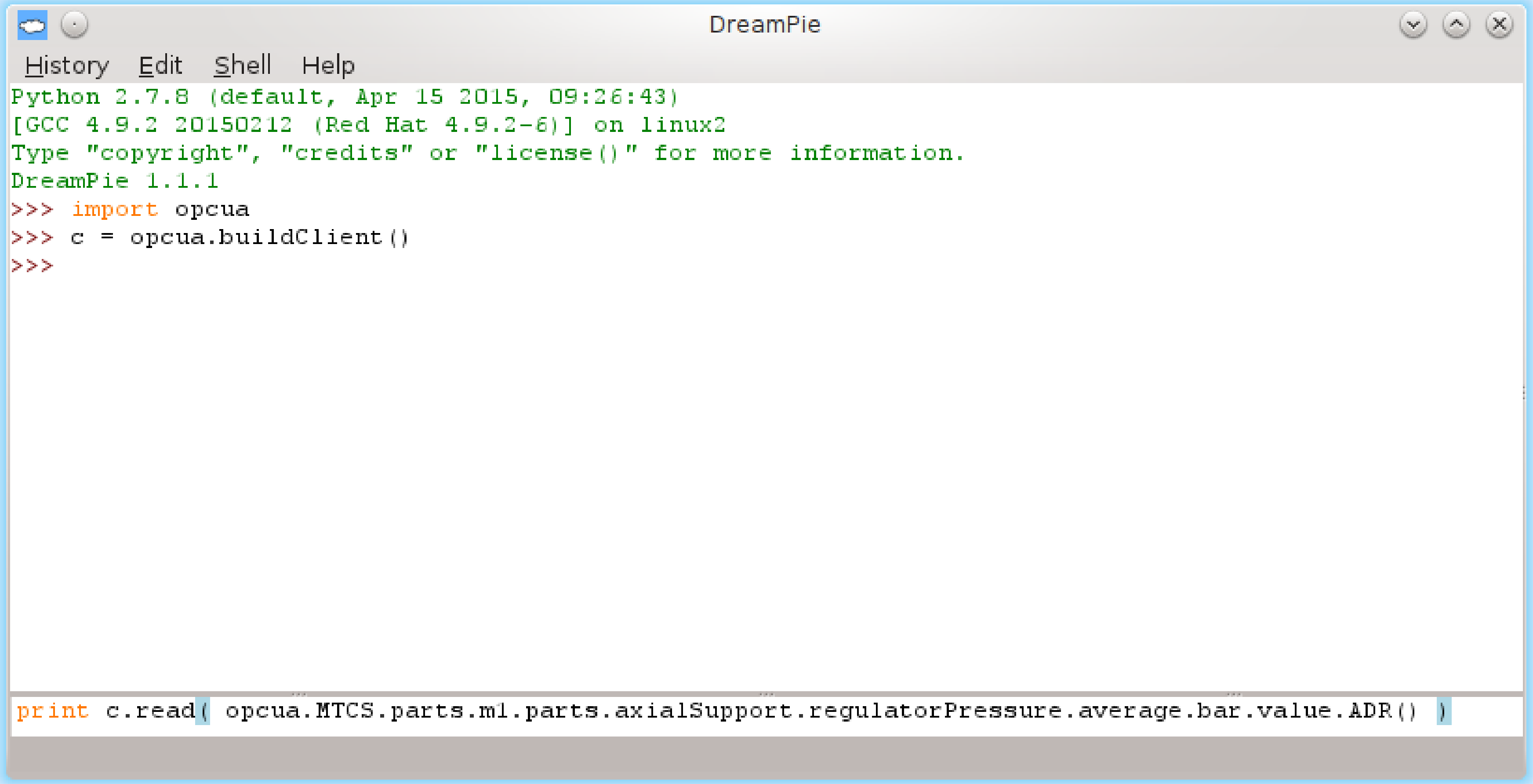
```
ADR
INFO
actualStatus
unit
value
```

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Software design

- Generated Python code (client side)
  - Based on our OPC UA library "UAF": http://github.com/uaf/uaf

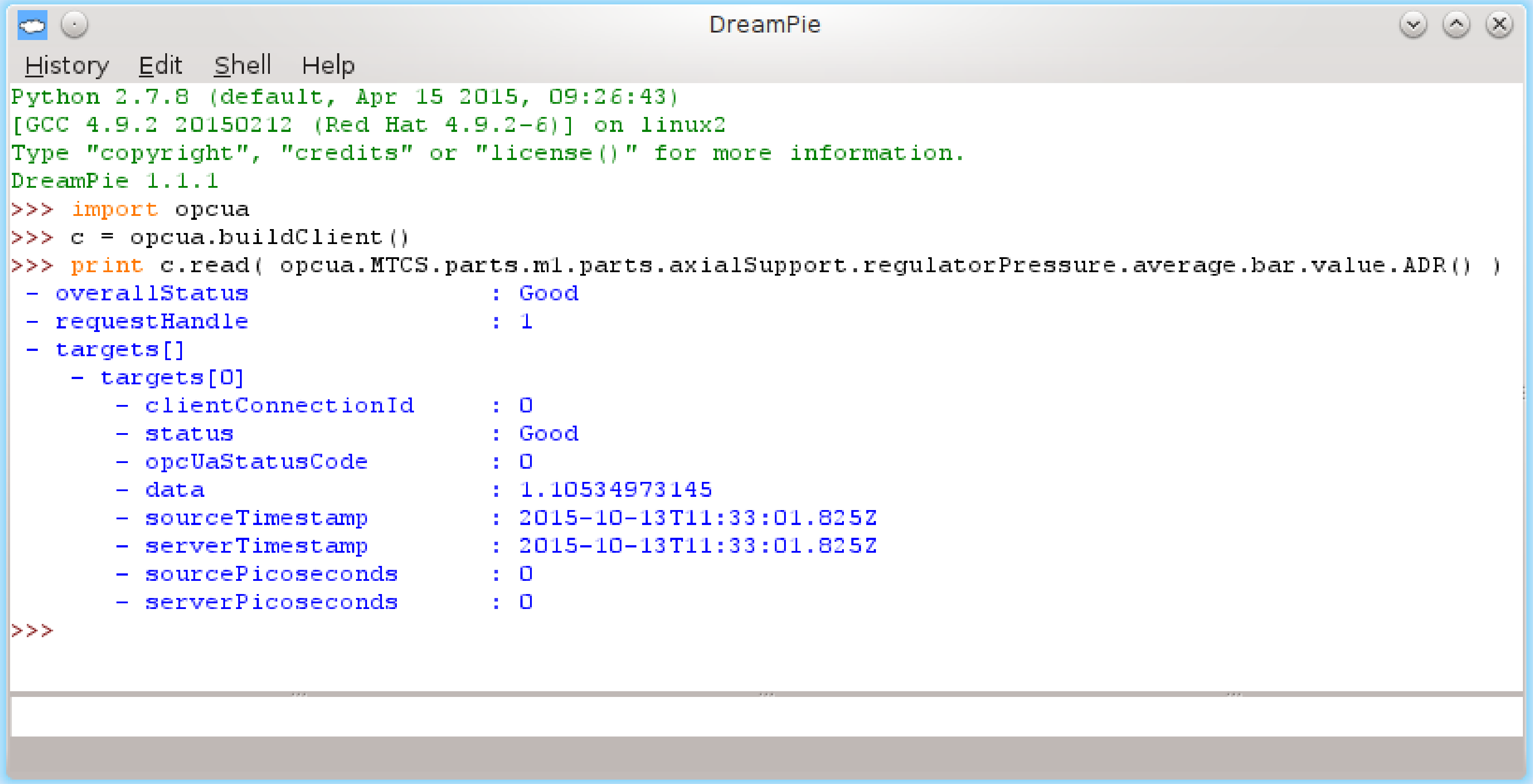


```
Python 2.7.8 (default, Apr 15 2015, 09:26:43)
[GCC 4.9.2 20150212 (Red Hat 4.9.2-6)] on linux2
Type "copyright", "credits" or "license()" for more information.
DreamPie 1.1.1
>>> import opcua
>>> c = opcua.buildClient()
>>>
```

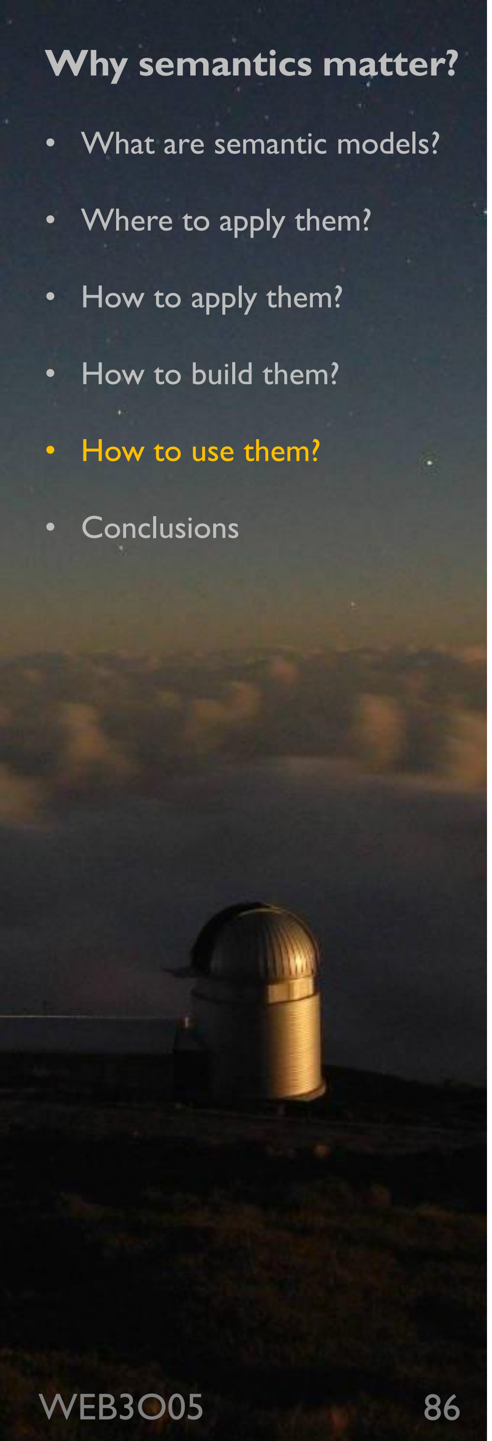
```
print c.read( opcua.MTCS.parts.m1.parts.axialSupport.regulatorPressure.average.bar.value.ADR() )
```

Software design

- Generated Python code (client side)
  – Based on our OPC UA library “UAF”: http://github.com/uaf/uaf



```
Python 2.7.8 (default, Apr 15 2015, 09:26:43)
[GCC 4.9.2 20150212 (Red Hat 4.9.2-6)] on linux2
Type "copyright", "credits" or "license()" for more information.
DreamPie 1.1.1
>>> import opcua
>>> c = opcua.buildClient()
>>> print c.read( opcua.MTCS.parts.m1.parts.axialSupport.regulatorPressure.average.bar.value.ADR() )
 - overallStatus                : Good
 - requestHandle                : 1
 - targets[]
    - targets[0]
       - clientConnectionId     : 0
       - status                 : Good
       - opcUaStatusCode        : 0
       - data                   : 1.10534973145
       - sourceTimestamp        : 2015-10-13T11:33:01.825Z
       - serverTimestamp        : 2015-10-13T11:33:01.825Z
       - sourcePicoseconds      : 0
       - serverPicoseconds      : 0
>>>
```

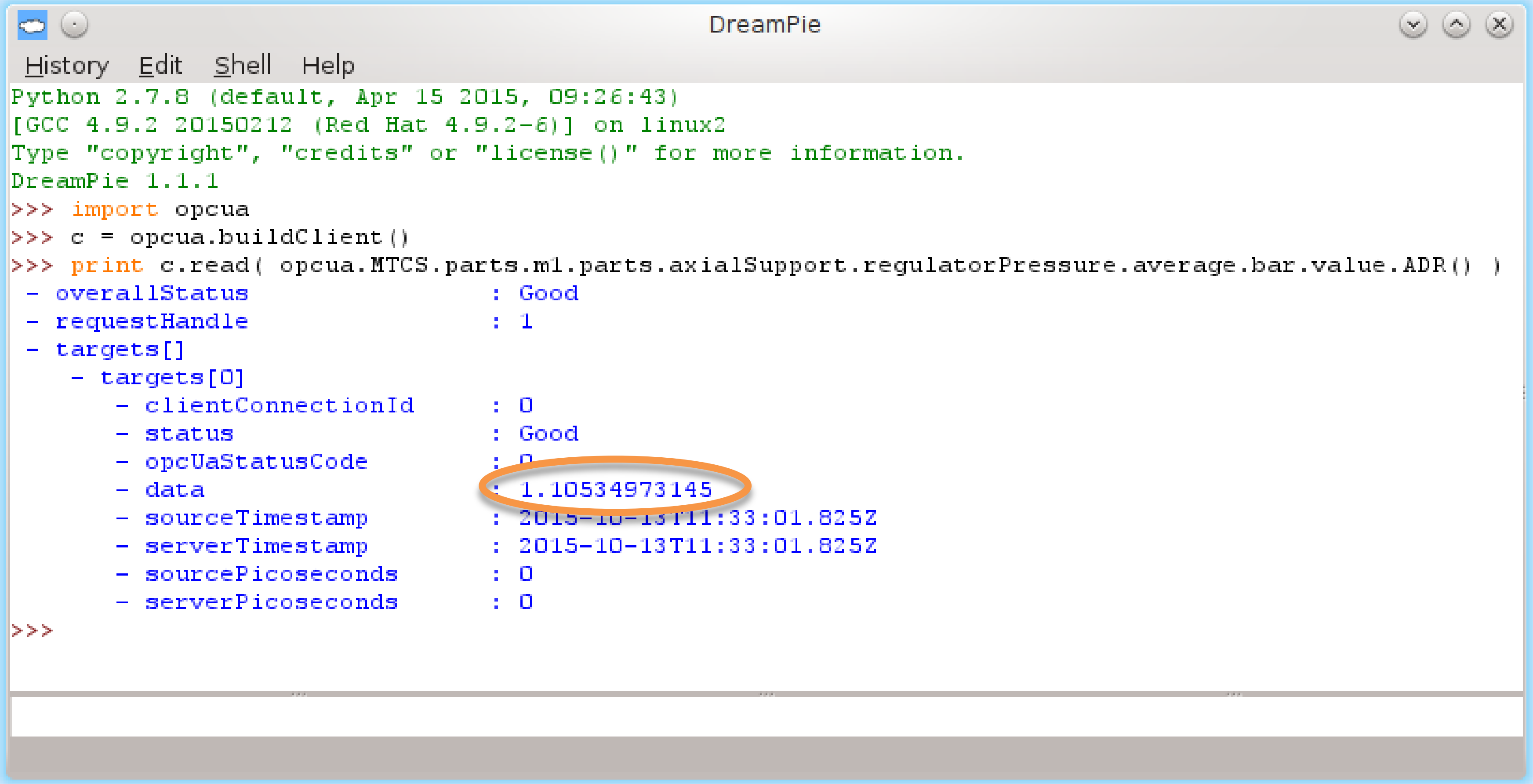


Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Software design

- Generated Python code (client side)
  - Based on our OPC UA library "UAF": http://github.com/uaf/uaf



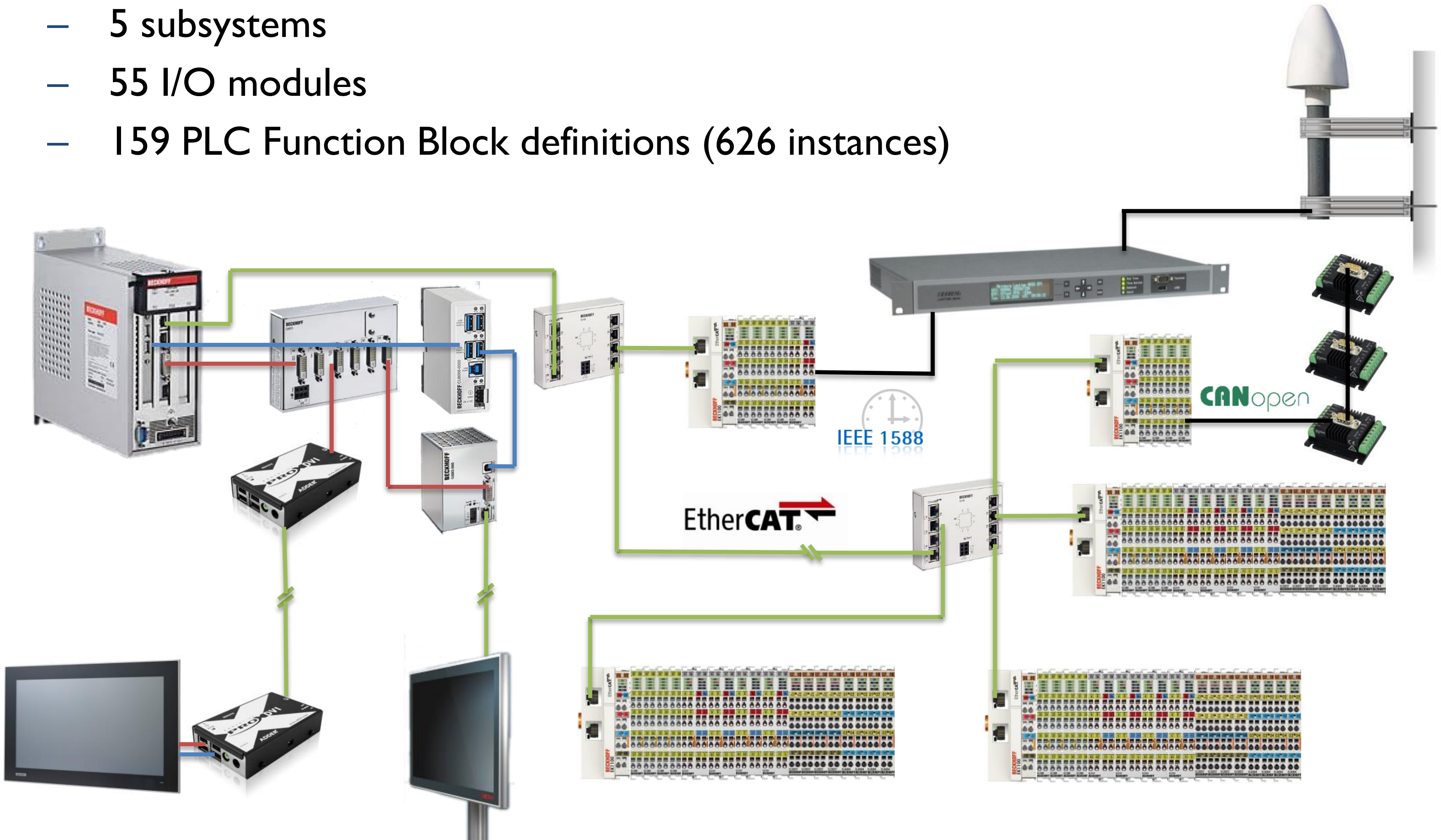
```
Python 2.7.8 (default, Apr 15 2015, 09:26:43)
[GCC 4.9.2 20150212 (Red Hat 4.9.2-6)] on linux2
Type "copyright", "credits" or "license()" for more information.
DreamPie 1.1.1
>>> import opcua
>>> c = opcua.buildClient()
>>> print c.read( opcua.MTCS.parts.m1.parts.axialSupport.regulatorPressure.average.bar.value.ADR() )
 - overallStatus                : Good
 - requestHandle                : 1
 - targets[]
    - targets[0]
       - clientConnectionId     : 0
       - status                 : Good
       - opcUaStatusCode        : 0
       - data                   : 1.10534973145
       - sourceTimestamp        : 2015-10-13T11:33:01.825Z
       - serverTimestamp        : 2015-10-13T11:33:01.825Z
       - sourcePicoseconds      : 0
       - serverPicoseconds      : 0
>>>
```

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Results

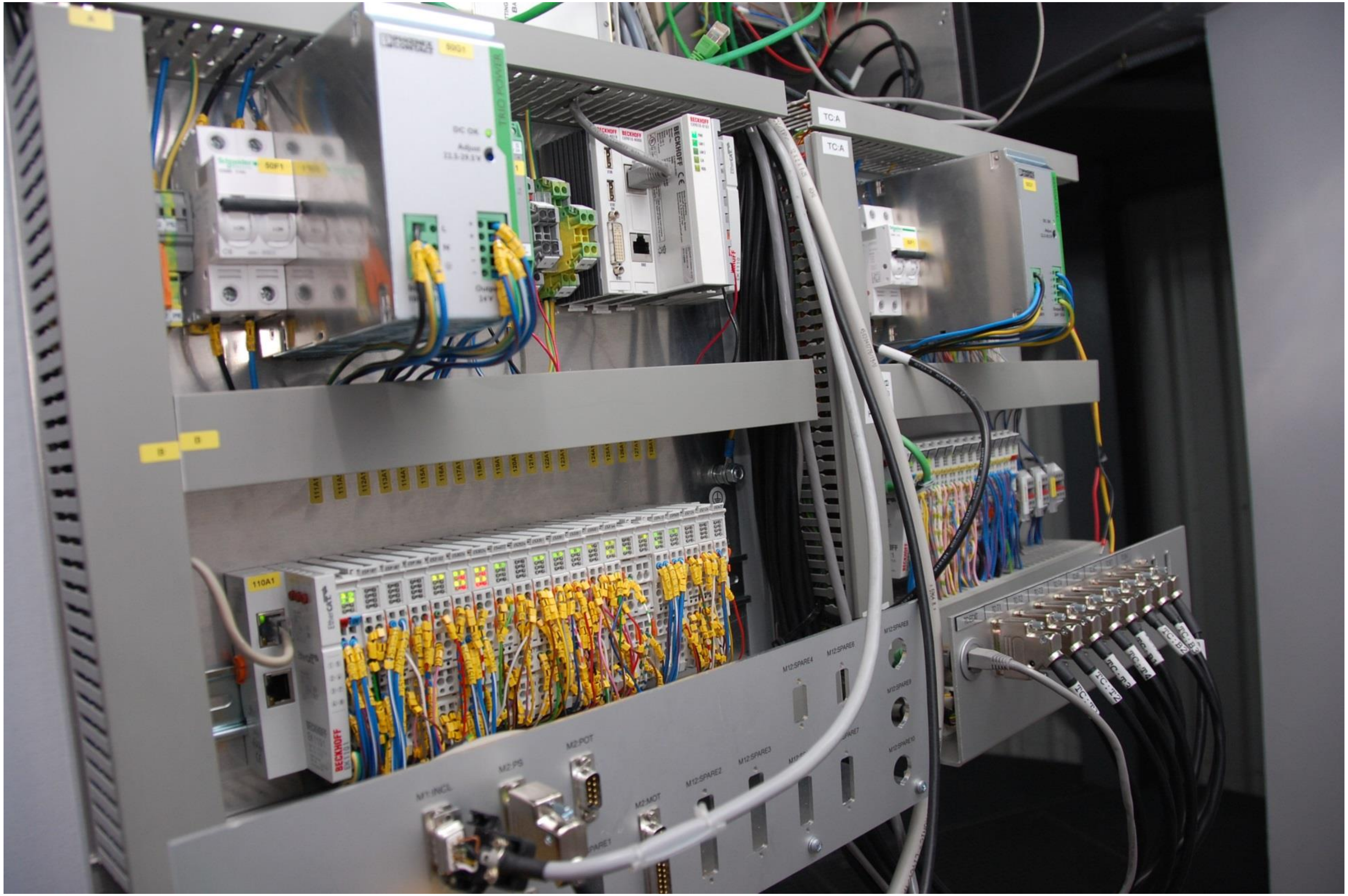
- Currently in operation:
  - 1 PLC
  - 5 subsystems
  - 55 I/O modules
  - 159 PLC Function Block definitions (626 instances)



Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

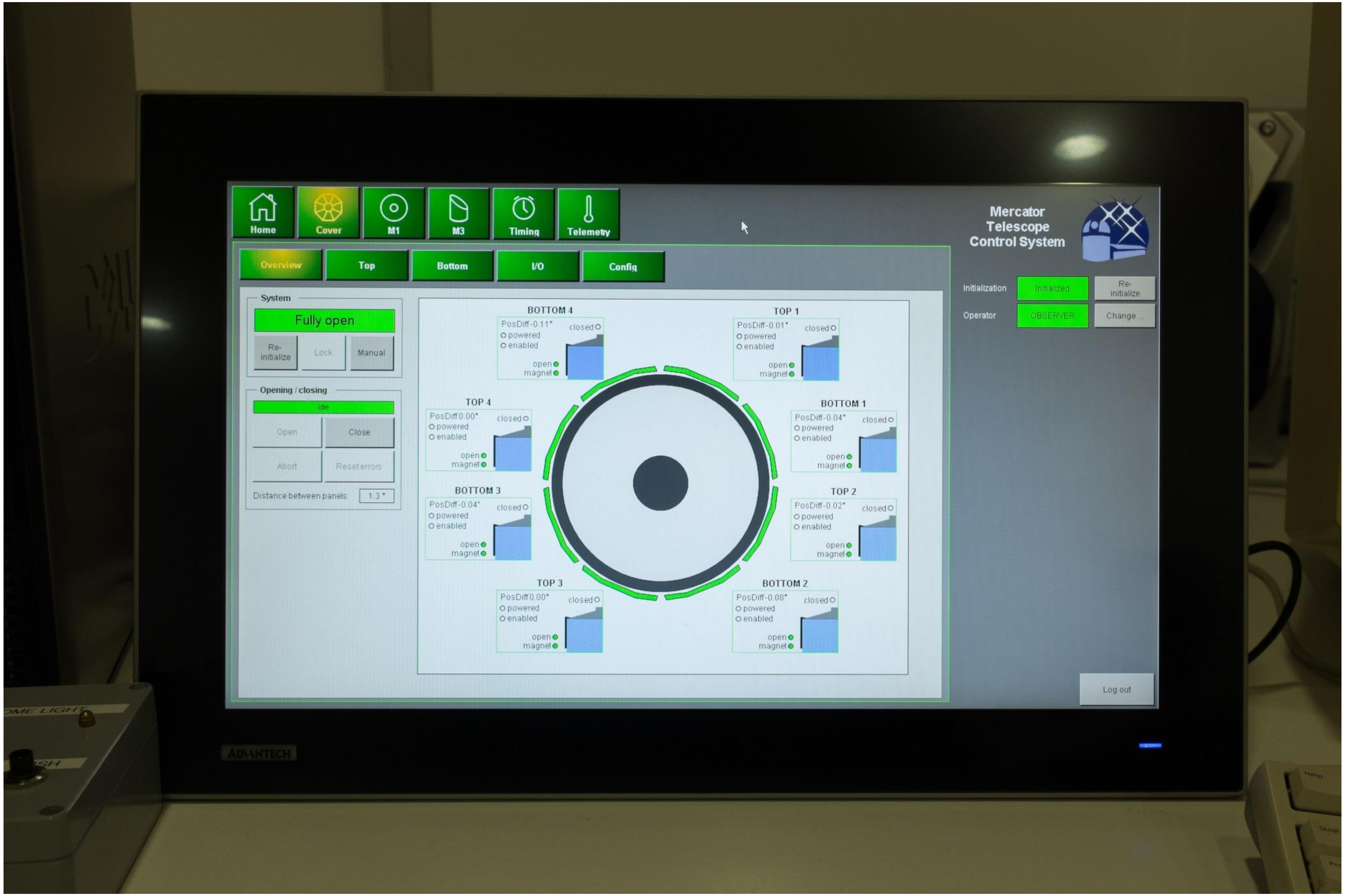
Results



Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Results



Mercator Telescope Control System

Initialization: Initialized

Operator: OBSERVER

System: Fully open

Opening / closing: Idle

Distance between panels: 1.3 °

BOTTOM 4: PosDiff -0.11° ○ powered ○ enabled closed ○ open ● magnet ●

TOP 1: PosDiff -0.01° ○ powered ○ enabled closed ○ open ● magnet ●

TOP 4: PosDiff 0.00° ○ powered ○ enabled closed ○ open ● magnet ●

BOTTOM 1: PosDiff -0.04° ○ powered ○ enabled closed ○ open ● magnet ●

BOTTOM 3: PosDiff -0.04° ○ powered ○ enabled closed ○ open ● magnet ●

TOP 2: PosDiff -0.02° ○ powered ○ enabled closed ○ open ● magnet ●

TOP 3: PosDiff 0.00° ○ powered ○ enabled closed ○ open ● magnet ●

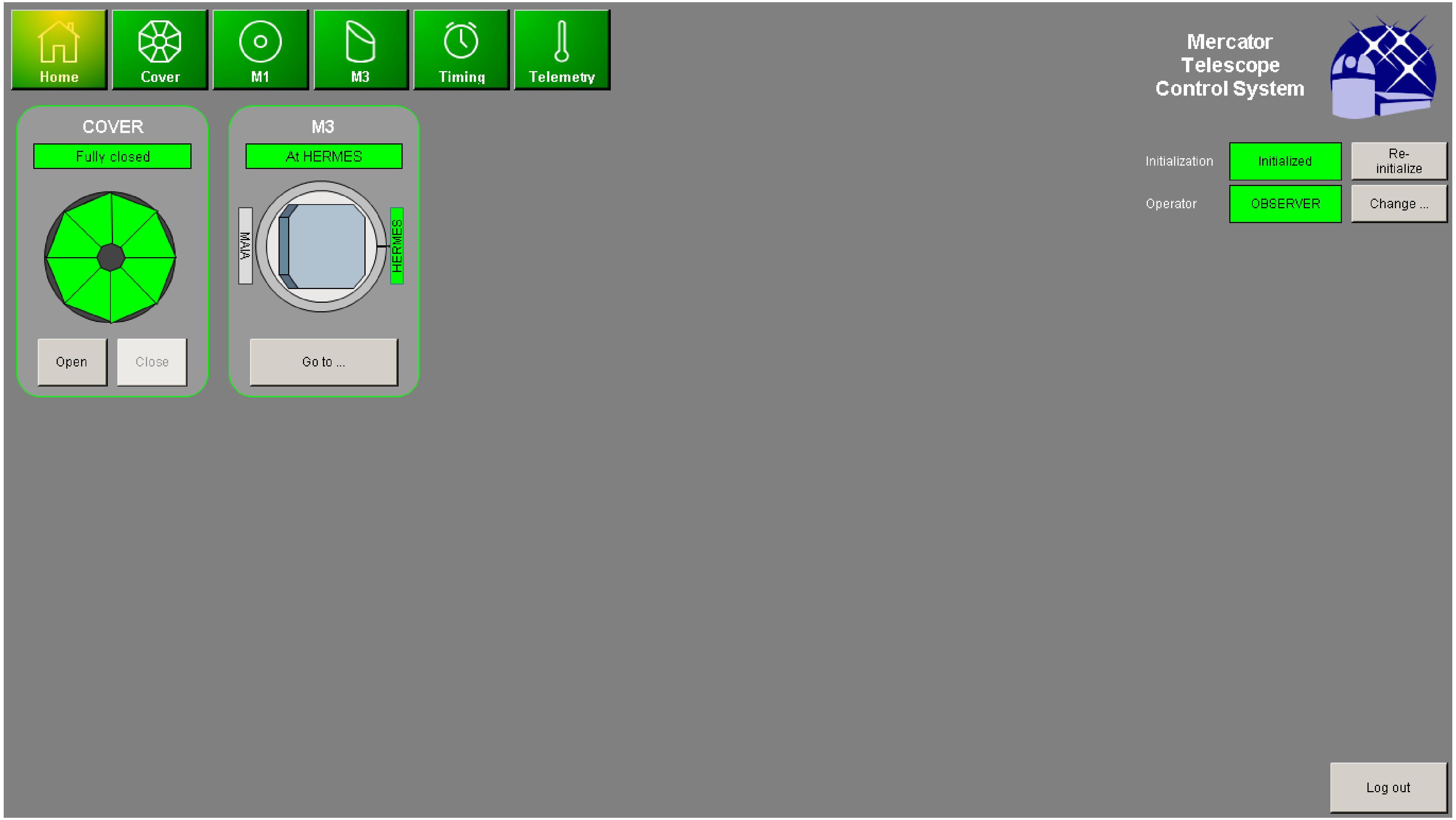
BOTTOM 2: PosDiff -0.08° ○ powered ○ enabled closed ○ open ● magnet ●

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Results

- User Interface (HMI) running **on** the PLC



COVER
Fully closed

M3
At HERMES

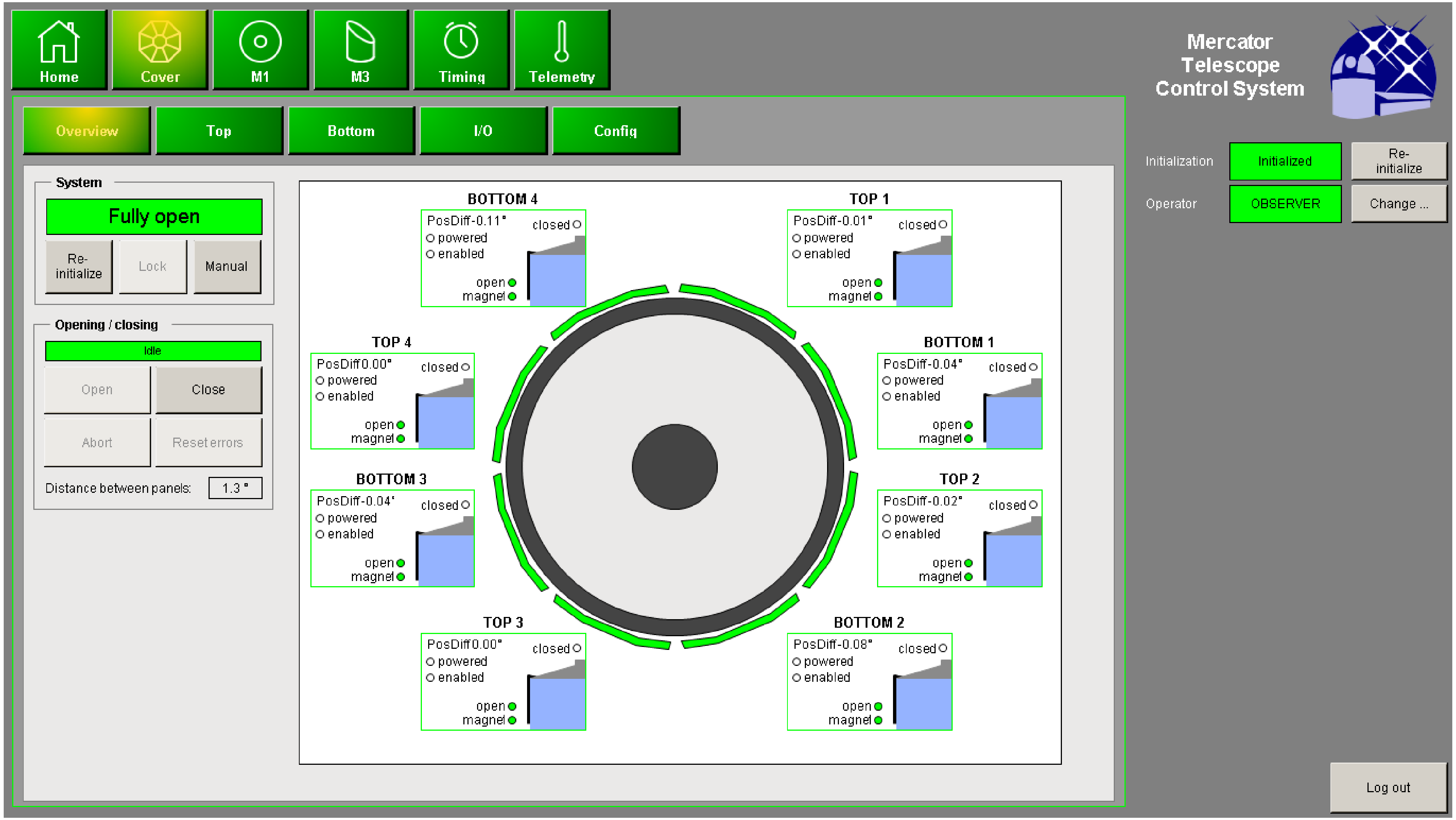
Initialization: Initialized
Operator: OBSERVER

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Results

- User Interface (HMI) running **on** the PLC



Mercator Telescope Control System

Initialization: Initialized

Operator: OBSERVER

System: Fully open

Opening / closing: Idle

Distance between panels: 1.3 °

BOTTOM 4
PosDiff-0.11° closed ○
○ powered
○ enabled
open ●
magnet ●

TOP 1
PosDiff-0.01° closed ○
○ powered
○ enabled
open ●
magnet ●

TOP 4
PosDiff0.00° closed ○
○ powered
○ enabled
open ●
magnet ●

BOTTOM 1
PosDiff-0.04° closed ○
○ powered
○ enabled
open ●
magnet ●

BOTTOM 3
PosDiff-0.04° closed ○
○ powered
○ enabled
open ●
magnet ●

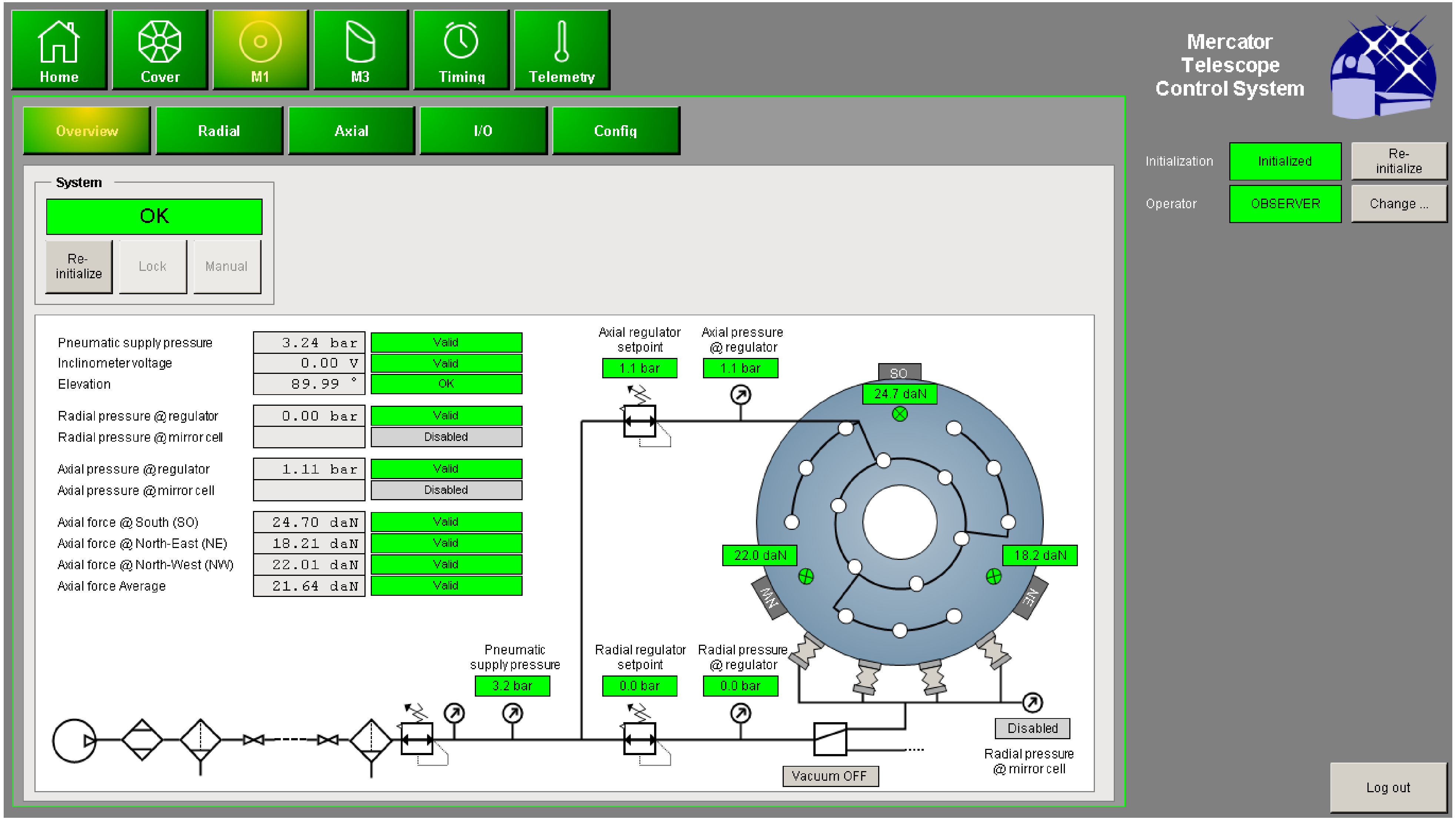
TOP 2
PosDiff-0.02° closed ○
○ powered
○ enabled
open ●
magnet ●

TOP 3
PosDiff0.00° closed ○
○ powered
○ enabled
open ●
magnet ●

BOTTOM 2
PosDiff-0.08° closed ○
○ powered
○ enabled
open ●
magnet ●

Results

- User Interface (HMI) running **on** the PLC


Home
Cover
M1
M3
Timing
Telemetry
Mercator Telescope Control System
Overview
Radial
Axial
I/O
Config
Re-initialize
Change ...
Log out
System
Re-initialize
Lock
Manual
Axial regulator setpoint
Axial pressure @ regulator
SO
NW
NE
Pneumatic supply pressure
Radial regulator setpoint
Radial pressure @ regulator
Radial pressure @ mirror cell
Vacuum OFF

Initialization: Initialized

Operator: OBSERVER

System: OK

Pneumatic supply pressure	3.24 bar	Valid
Inclinometer voltage	0.00 V	Valid
Elevation	89.99 °	OK
Radial pressure @ regulator	0.00 bar	Valid
Radial pressure @ mirror cell		Disabled
Axial pressure @ regulator	1.11 bar	Valid
Axial pressure @ mirror cell		Disabled
Axial force @ South (SO)	24.70 daN	Valid
Axial force @ North-East (NE)	18.21 daN	Valid
Axial force @ North-West (NW)	22.01 daN	Valid
Axial force Average	21.64 daN	Valid

Axial regulator setpoint: 1.1 bar

Axial pressure @ regulator: 1.1 bar

SO: 24.7 daN

NW: 22.0 daN

NE: 18.2 daN

Pneumatic supply pressure: 3.2 bar

Radial regulator setpoint: 0.0 bar

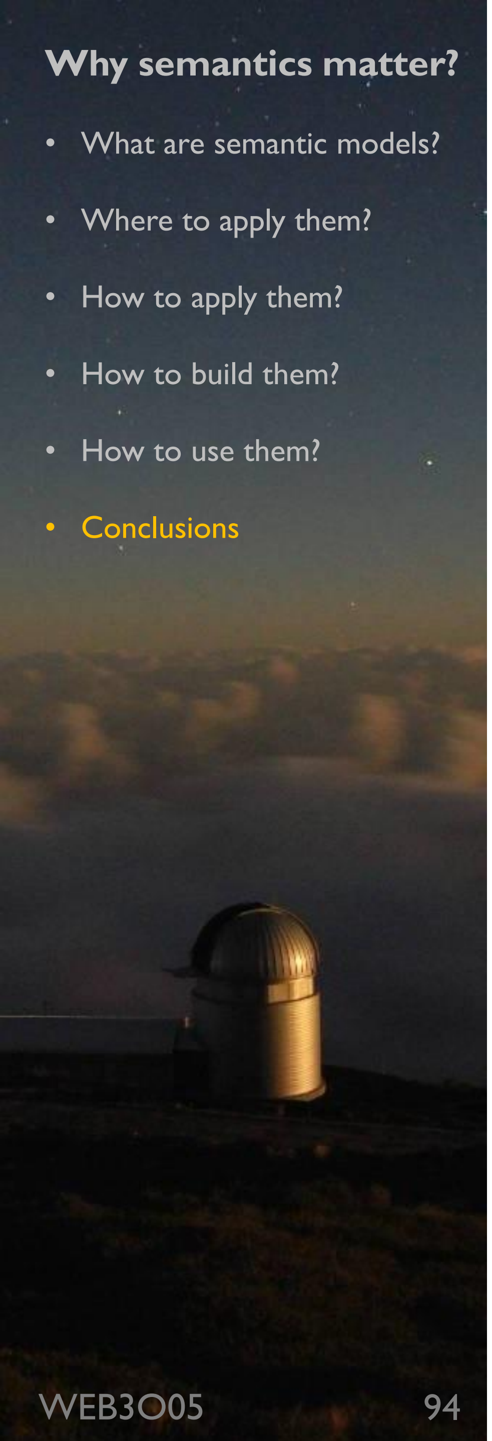
Radial pressure @ regulator: 0.0 bar

Radial pressure @ mirror cell: Disabled

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Conclusions



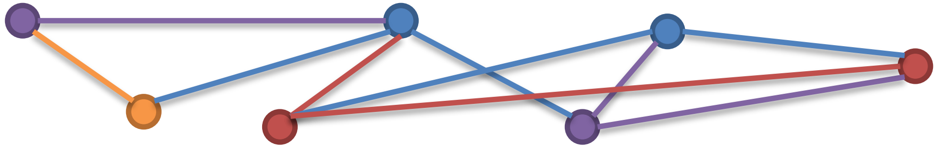
Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

So, why semantics matter?

So, why semantics matter?

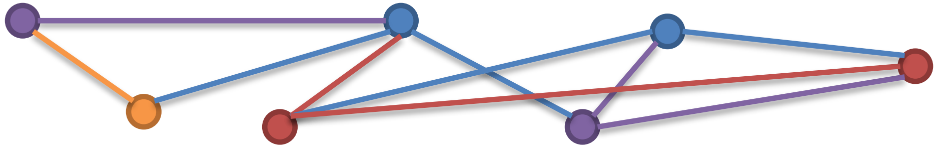
1. Because every piece of information is just one query “away”



➔ **organize, integrate, browse, find (query)** information

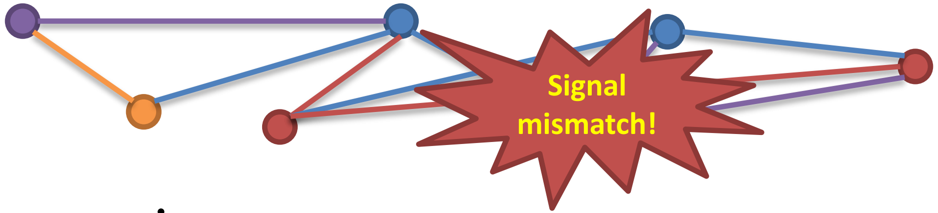
So, why semantics matter?

1. Because every piece of information is just one query "away"



➔ **organize, integrate, browse, find (query)** information

2. Because well defined semantics allow model verification



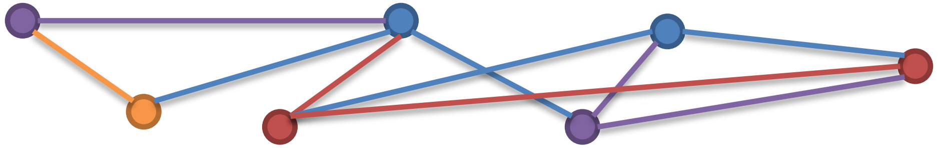
➔ **verify** information

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

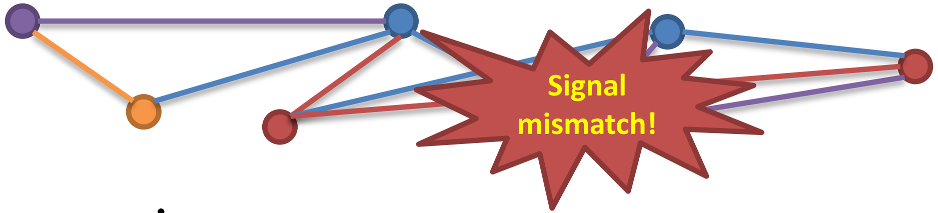
So, why semantics matter?

1. Because every piece of information is just one query "away"



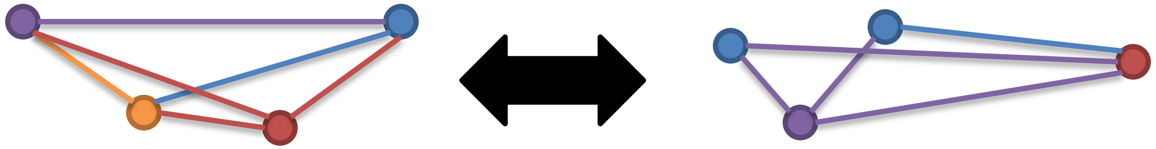
➔ **organize, integrate, browse, find (query)** information

2. Because well defined semantics allow model verification



➔ **verify** information

3. Because they're a key enabling technology for future "smart" systems



➔ **share** information

Thanks!

Any questions?

wim.pessemier@ster.kuleuven.be