Reflective Language for Communications Systems

WInnComm 2014

Vince Kovarik

PrismTech

Disclaimer

The following presentation presents personal work performed by Vince Kovarik.

It does not reflect PrismTech views, products or capabilities.

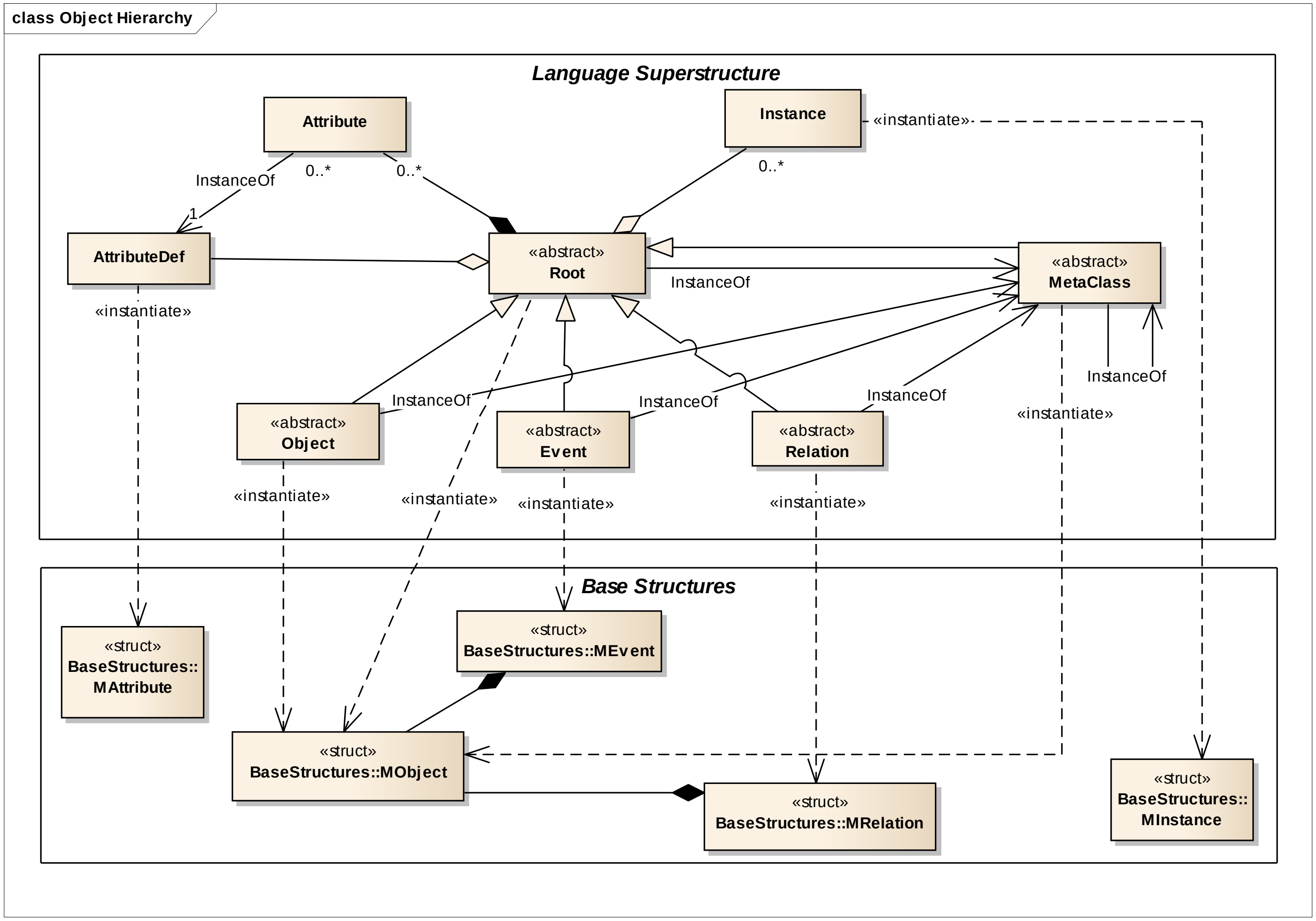
Origins

- **Prior experience in knowledge representation and reasoning, e.g. Frames, Conceptual Dependency, NETL, Cyc, OWL, etc.**
  - Expressive representation or reasoning capability but often limited in domain or scope.
- **Experience with object-oriented languages, e.g. LISP/Flavors, LOOPS, CLOS, Smalltalk, Simula, C++, Java, etc.**
  - Provides the association of state (variables) and behavior (methods) within the concept of an object but limited to current value.
- **Brittleness in existing programming languages, including OO languages.**
  - Weak or no built-in representation of context

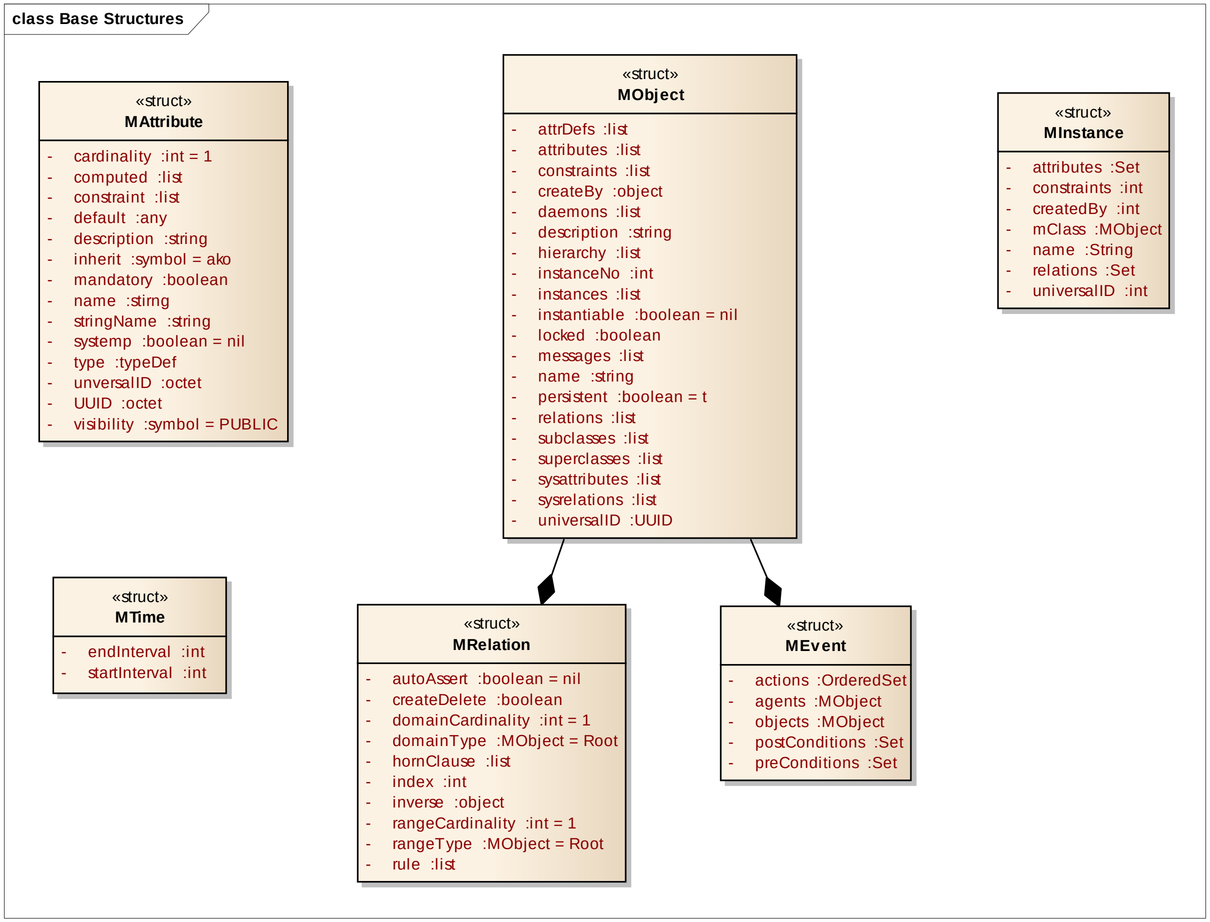
Basic Tenets/Objectives

- **Object-Oriented language paradigm but extended to represent relationships and events/actions as concrete entities.**
- **Incorporate a base set of objects, relations and events and common semantic interpretations within the base system.**
- **Prototype language as an extension of existing OO Language.**
- **Evolutionary optimization of low-level structural representation to improve performance.**
- **Provide capability to link to libraries for special purpose code, device drivers, etc.**

Language Superstructure



Base Structures



class Base Structures

«struct»
MAttribute

- cardinality :int = 1
- computed :list
- constraint :list
- default :any
- description :string
- inherit :symbol = ako
- mandatory :boolean
- name :stirng
- stringName :string
- systemp :boolean = nil
- type :typeDef
- unversalID :octet
- UUID :octet
- visibility :symbol = PUBLIC

«struct»
MObject

- attrDefs :list
- attributes :list
- constraints :list
- createBy :object
- daemons :list
- description :string
- hierarchy :list
- instanceNo :int
- instances :list
- instantiable :boolean = nil
- locked :boolean
- messages :list
- name :string
- persistent :boolean = t
- relations :list
- subclasses :list
- superclasses :list
- sysattributes :list
- sysrelations :list
- universalID :UUID

«struct»
MInstance

- attributes :Set
- constraints :int
- createdBy :int
- mClass :MObject
- name :String
- relations :Set
- universalID :int

«struct»
MTime

- endInterval :int
- startInterval :int

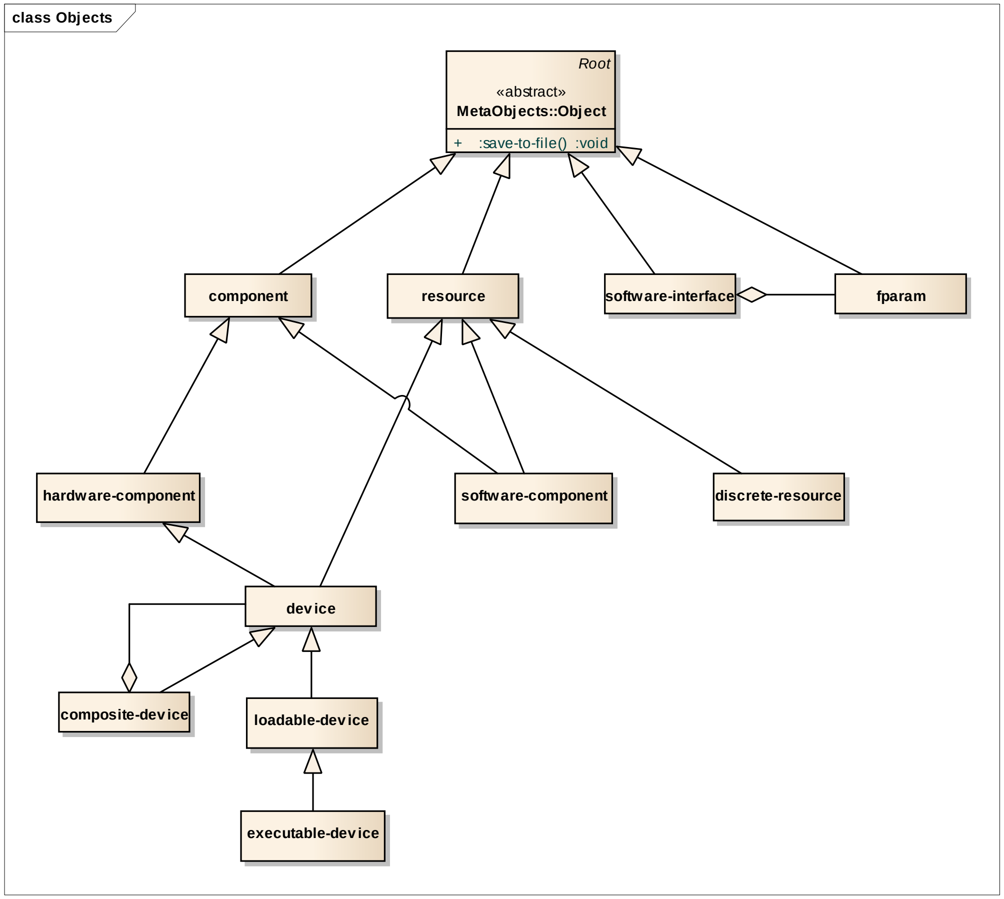
«struct»
MRelation

- autoAssert :boolean = nil
- createDelete :boolean
- domainCardinality :int = 1
- domainType :MObject = Root
- hornClause :list
- index :int
- inverse :object
- rangeCardinality :int = 1
- rangeType :MObject = Root
- rule :list

«struct»
MEvent

- actions :OrderedSet
- agents :MObject
- objects :MObject
- postConditions :Set
- preConditions :Set

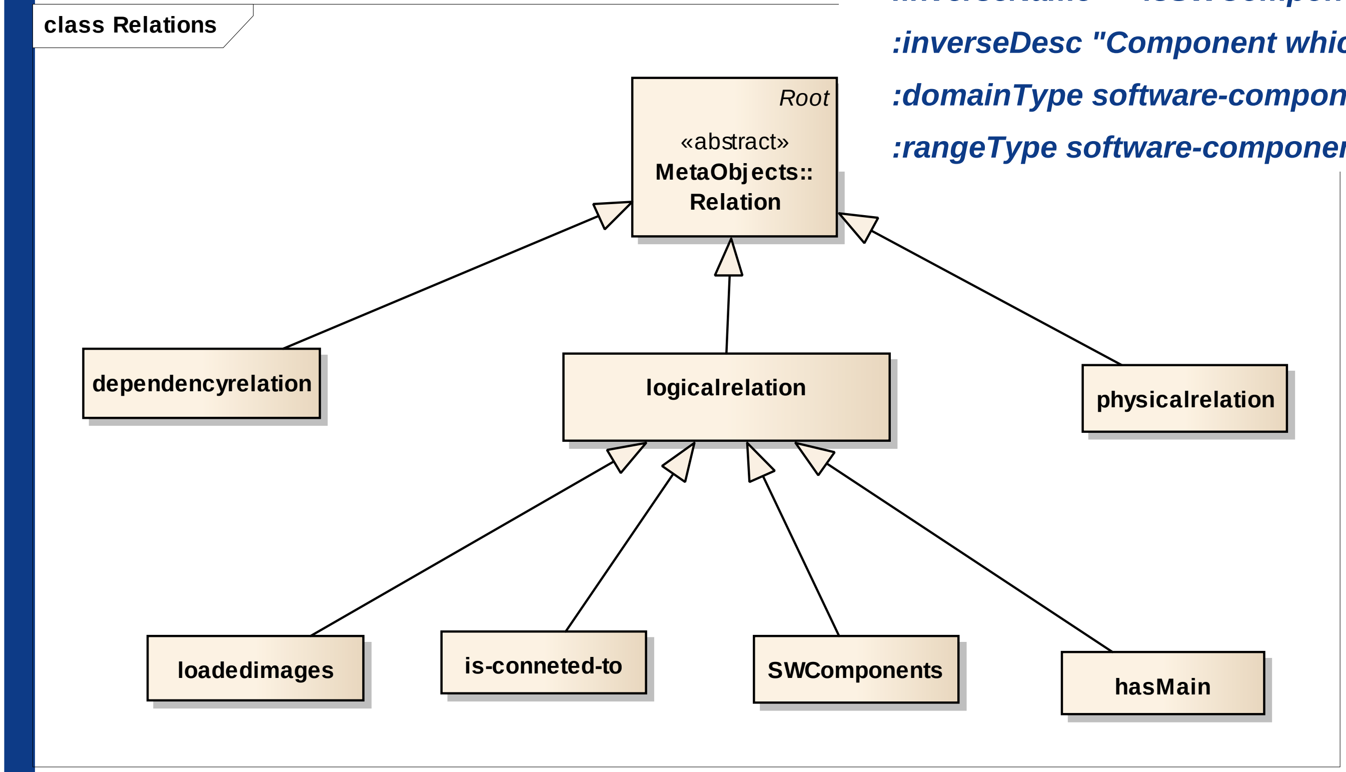
Base Object Hierarchy



```
(def-class resource object
   :name "Resource"
   :instantiable nil
   :description "A physical or logical
     element that is required or used by
     another element to provide some
     capability or function."
   :attributes (
         (id
            :cardinality 1
            :description "A unique identifier"
         )))
(def-method (resource :start) ()
   ; This would link to a .so or .dll library
     implementing the HW/Resource
     interface and logic
   (format t "Processing started for ~a~%"
     self))
```

Base Relation Hierarchy

```
(defrelation SWComponents logicalrelation
   :name "->hasSWComponents->"
   :description "Sub-components of this component"
   :inverse SwComponentOf
   :inverseName "->isSWComponentOf->"
   :inverseDesc "Component which this component is a part of"
   :domainType software-component
   :rangeType software-component)
```

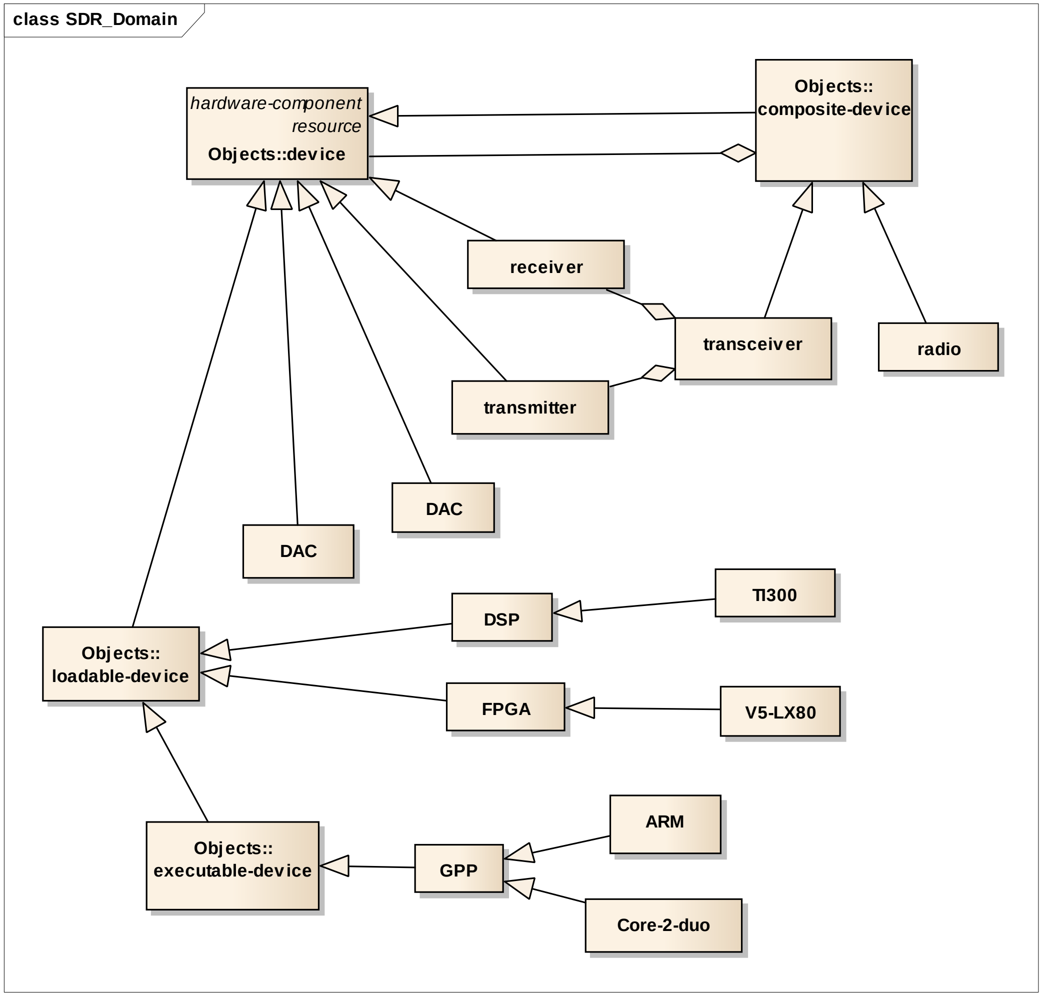


Driving the future of radio communications and systems worldwide

version 2.0

So, what does any of this have to do with communications systems?

Radio Domain Objects



- **Radio hardware components are defined as subclasses of a device.**
- **The composite-device provides a simple but effective characterization of a hierarchical collection of components.**

Define Software Components

```
(setf swc-1
   (tell 'software-component :new
      :props '(
         (name "Math Library")
         (size 800000)
         (language "C")
         )))

(setf swc-2
   (tell 'software-component :new
      :props '(
         (name "Tracking Main")
         (size 2000)
         (language "C")
         )))

(setf swc-3
   (tell 'software-component :new
      :props '(
         (name "IO Library")
         (size 3400)
         (language "ASM")
         )))

(setf swc-4
   (tell 'software-component :new
      :props '(
         (name "Rate 1/2 Viterbi Decoder")
         (size 3000)
         (language "VHDL")
         )))
```

Software Component Definition

```
(tell loaddev-1 :load swc-4)
(tell loaddev-1 :describe)
BeginInstance=SOFTWARE-COMPONENT-4
  InstanceOf->SOFTWARE-COMPONENT
         Name="SOFTWARE-COMPONENT-4"
Attributes:
       NAME="Rate 1/2 Viterbi Decoder"
       SIZE=3000
       LANGUAGE="VHDL"
 Relations:
      LOADEDON->(V5-LX80-1)
  EndInstance=SOFTWARE-COMPONENT-4
```

Define Software Components

```
(tell swc-2 :assert 'swcomponents swc-1)
(tell swc-2 :assert 'swcomponents swc-3)

(setf testapp
    (tell 'application :new
        :props '(
            (name "Tracking Application")
            )))

(tell testapp :assert 'hasMain swc-2)
```

Define Software Components

(tell testapp :describe)

```
BeginInstance=APPLICATION-1
 InstanceOf->APPLICATION
        Name="APPLICATION-1"
Attributes:
       NAME="Tracking Application"
 Relations:
      HASMAIN->(SOFTWARE-COMPONENT-2)
 EndInstance=APPLICATION-1
```

Software Application Main

(tell swc-2 :describe)

```
BeginInstance=SOFTWARE-COMPONENT-2
  InstanceOf->SOFTWARE-COMPONENT
         Name="SOFTWARE-COMPONENT-2"
Attributes:
       NAME="Tracking Main"
       SIZE=2000
       LANGUAGE="C"
 Relations:
      SWCOMPONENTS->(SOFTWARE-COMPONENT-1 SOFTWARE-
     COMPONENT-3)
      ISMAINOF->(APPLICATION-1)
  EndInstance=SOFTWARE-COMPONENT-2
```

Hardware Component Definition

```
(def-class V5-LX80 FPGA
   :name "Virtex 5 LX80"
   :description "A model of FPGA"
   :instantiable t
   :attributes (
      (numgates
         :default 60000)
      (maxClockRate
         :default 240)
      (maxImage
         :default 10000)
      (availImage
         :default 10000)
      ))
```

```
(setf loaddev-1
   (tell 'V5-LX80 :new))

(tell loaddev-1 :describe)
BeginInstance=V5-LX80-1
  InstanceOf->V5-LX80
         Name="V5-LX80-1"
Attributes:
 Relations:
  EndInstance=V5-LX80-1

(tell loaddev-1 :get 'maximage)
10000
```

“Defining” a Radio

```
(def-class lmr-radio radio
   :name "Land Mobile Radio"
   :description "A trunked, handheld radio used by public safety and first responders."
   :instantiable t
   :attributes (
      (minFreq
         :description "Minimum operating frequency"
         :cardinality 1
         :default 250.1
         :units MHz)
      (maxFreq
         :description "Maximum operating frequency"
         :cardinality 1
         :default 338.0
         :units MHz)))

(tell 'lmr-radio :assert 'parts '(arm TI300 transceiver))
(tell 'receiver :assert 'parts 'ad-24)
(tell 'transmitter :assert 'parts 'class-c-PA)
```

“Defining” a Radio

BeginClass:LMR-RADIO

SuperClass->(RADIO)

ClassType=CLASS

ClassName=*"Land Mobile Radio"*

isLocked=NIL

isInstantiable=T

isPersistent=T

InstancesCount=0

AttributeDefinitions:

MAXFREQ:

Description=*"Maximum operating frequency"*

Cardinality=1

DefaultValue=338.0

InheritRel=AKO

DataType=NIL

MINFREQ:

Description=*"Minimum operating frequency"*

Cardinality=1

DefaultValue=250.1

InheritRel=AKO

DataType=NIL

Attributes:

Relations:

PARTS->(ARM TI300 TRANSCEIVER)

EndClass:LMR-RADIO

Defining a Radio

```
(tell 'lmr-radio :print-hierarchy)
PARTS hierarchy of:
LMR-RADIO - "Land Mobile Radio"
..ARM - "ARM Processor"
..TI300 - "Texas Instruments 300 DSP"
..TRANSCEIVER - "Transceiver"
....V5-LX80 - "Virtex 5 LX80"
....ARM - "ARM Processor"
....TI300 - "Texas Instruments 300 DSP"
....TRANSMITTER - "Transmitter"
......CLASS-C-PA - "Class C Power Amplifier"
....RECEIVER - "Receiver"
......AD-24 - "24 bit ADC"
```

Instantiating a Radio

(setf lmr1 (tell 'lmr-radio :new))

```
BeginClass:LMR-RADIO
   SuperClass->(RADIO)
     ClassType=CLASS
     ClassName="Land Mobile Radio"
      isLocked=NIL
isInstantiable=T
  isPersistent=T
InstancesCount=1
AttributeDefinitions:
  MAXFREQ:
     Description="Maximum operating
       frequency"
     Cardinality=1
    DefaultValue=338.0
      InheritRel=AKO
        DataType=NIL
MINFREQ:
     Description="Minimum operating
       frequency"
     Cardinality=1
    DefaultValue=250.1
      InheritRel=AKO
        DataType=NIL
Attributes:
  LAST-INSTANCE=LMR-RADIO-1
Relations:
  PARTS->(ARM TI300 TRANSCEIVER)
EndClass:LMR-RADIO
```

Instantiating a Radio

```
(tell lmr1 :print-hierarchy)
PARTS hierarchy of:
LMR-RADIO-1 - "LMR-RADIO-1"
..ARM-1 - "ARM-1"
..TI300-1 - "TI300-1"
..TRANSCEIVER-1 - "TRANSCEIVER-1"
....V5-LX80-2 - "V5-LX80-2"
....ARM-2 - "ARM-2"
....TI300-2 - "TI300-2"
....TRANSMITTER-1 - "TRANSMITTER-1"
....RECEIVER-1 - "RECEIVER-1"
```

Summary

- **Objective is to extend the expressiveness of an executable language to incorporate:**
  - Objects, Relations and Events as first-class objects in the system.
  - Context – expressed as the set of relationships asserted for an object
    - Logical, physical, spatial, temporal
  - State – The value of the properties for an object.
    - Incorporate historical values enabling the ability to query an object about it's state at a given point in time or in relation to an event.
  - Inference and Reasoning – Representing heuristics as a set of conditions tied to assertions or action via relationships.
  - Hypotheses – The ability to define a set of assumptions or assertions that allow exploration of alternatives concurrently within the same knowledge space.