Ontology language To Support Description of Experiment Control System Semantics, Collaborative Knowledge-Base Design And Ontology Reuse

Vardan Gyurjyan
Thomas Jefferson National Accelerator Facility

Jefferson Lab

2000 CEBAF ~6 GeV beam delivered to experimental halls, exceeding machine design by 50 percent.
2000 JLab received contract to engineer and assemble the superconducting accelerator and to design and oversee installation of the helium refrigeration plant for the Spallation Neutron Source (SNS) at Oak Ridge National Lab.
2003 Groundbreaking data published on the shape of the proton.
2004 DOE approved "mission need" for JLab's 12 GeV Upgrade.
2005 JLab data revealed that the contribution of strange quarks to proton properties is small.
2005 Free-Electron Laser earned R&D 100 Award.
2006 Upgraded Free-Electron Laser surpassed 10kW design to achieve 14.2 kW in the infrared.
2007 Cryogenics Group wins prestigious White House award for energy-saving advancements.
2008 12 GeV Upgrade Project received Critical Decision-3 Approval from the U.S. Dept. of Energy.

Jefferson Lab
Thomas Jefferson National Accelerator Facility

Outline

- Experiment Control Domain
- Control System Conceptualization
- Ontology Model and Language Syntax
- State Machine Description
- Summary and Conclusions

Experiment Control Domain

- Real world physical components constitute experiment
- Channels are observables that describe physical components
- Collective values of these observables define the state of component
- States and state transitions define component specific behavior
- Rules describe behaviors of the component/components and are designed to achieve control objectives of the experiment

Control System Conceptualization

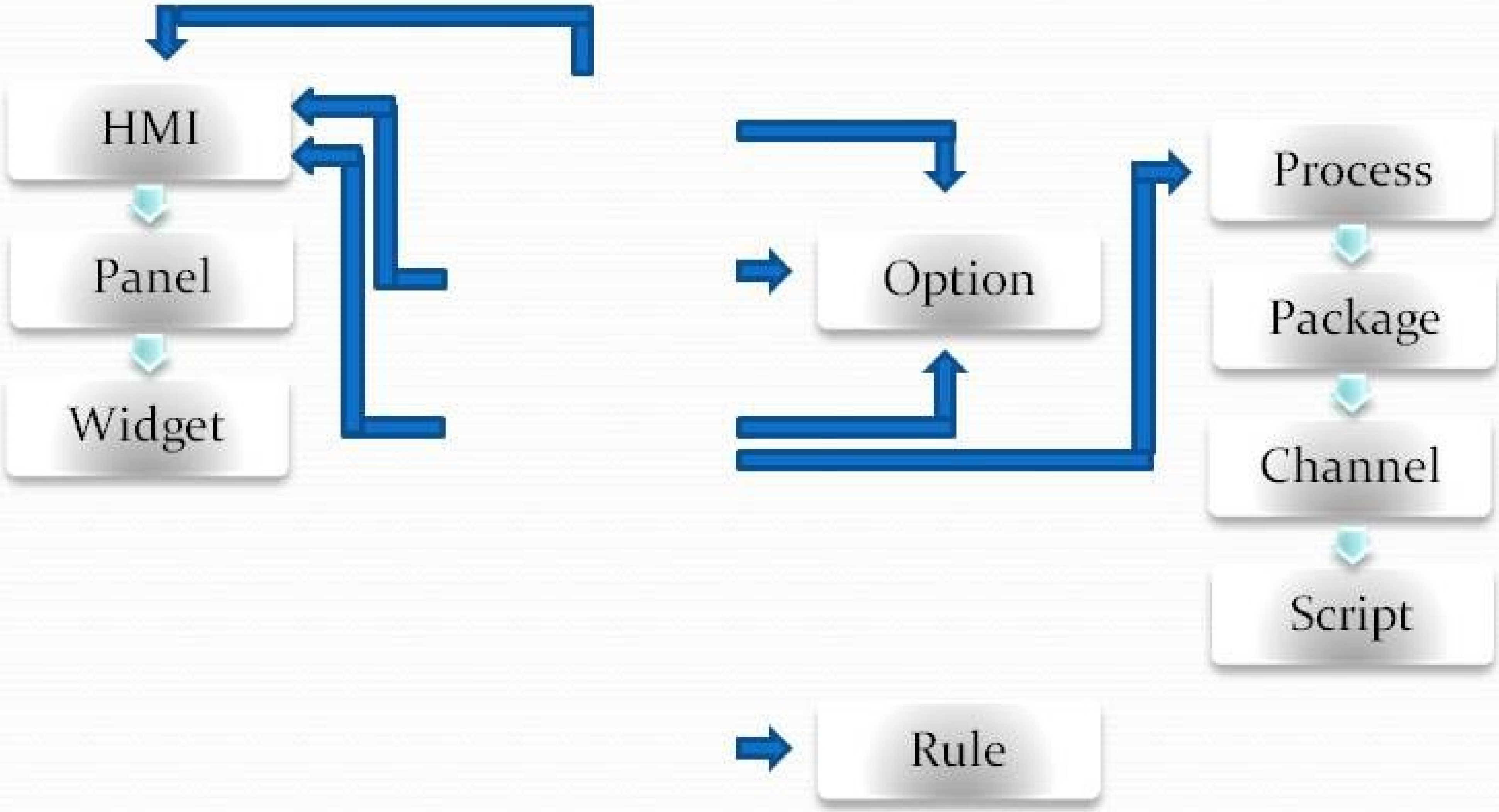
- Identify natural language terms that refer to concepts, relations among them and their attributes
- Concepts and their properties as a control system modeling primitive
- An experiment knowledge-base consists of instances of concepts and relationships between them

Relationships Between Concepts

- Taxonomic relations are used to describe further specialization of a concept.
  - **Component** *isSupervisor*
  - **Process** *isInitiator*

- Associative relations relate concept across the language structure.
  - **Component** *hasState*
  - **State** *achievedThrough*

Language Basic Concepts

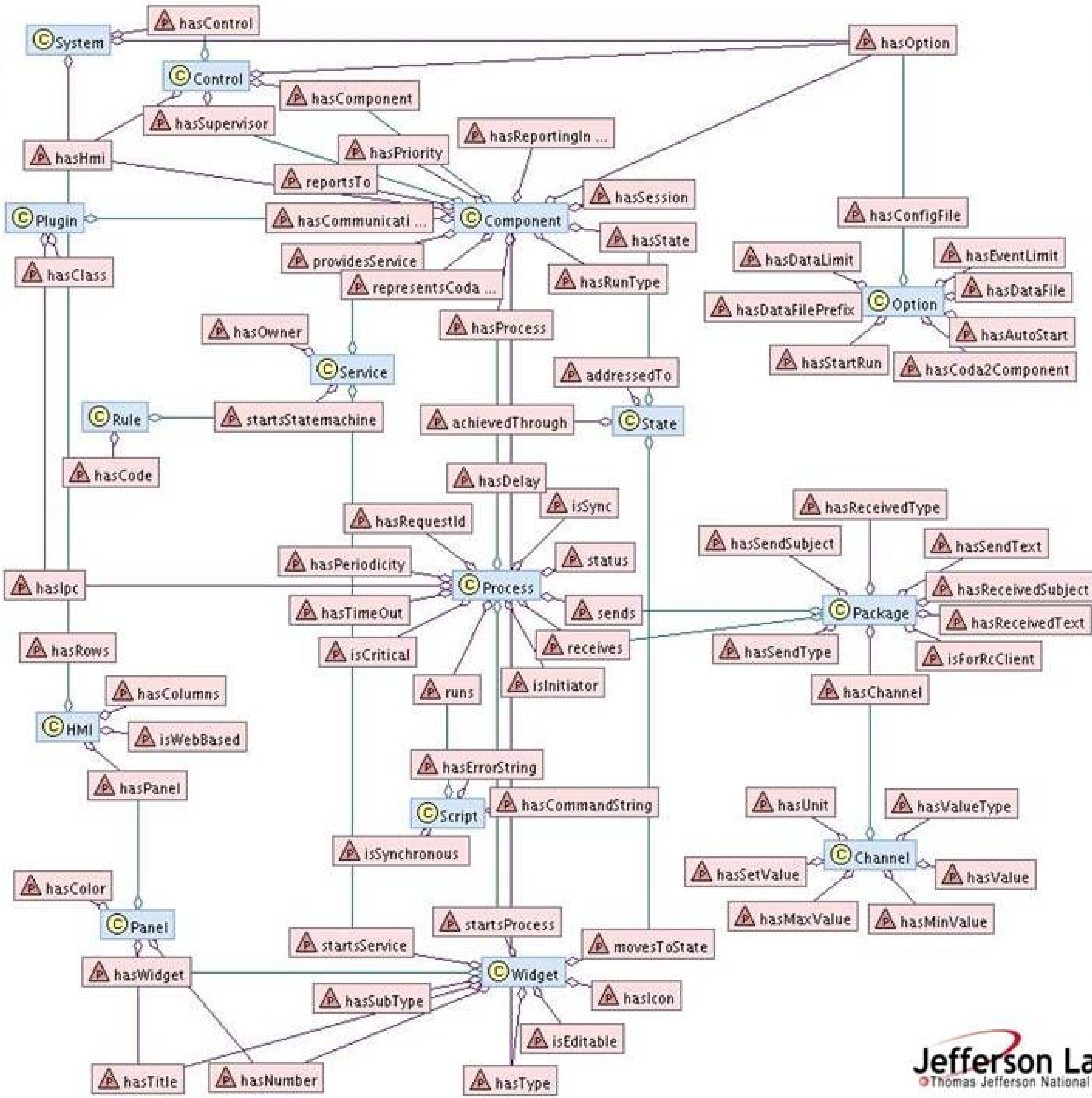


Language Model and Syntax

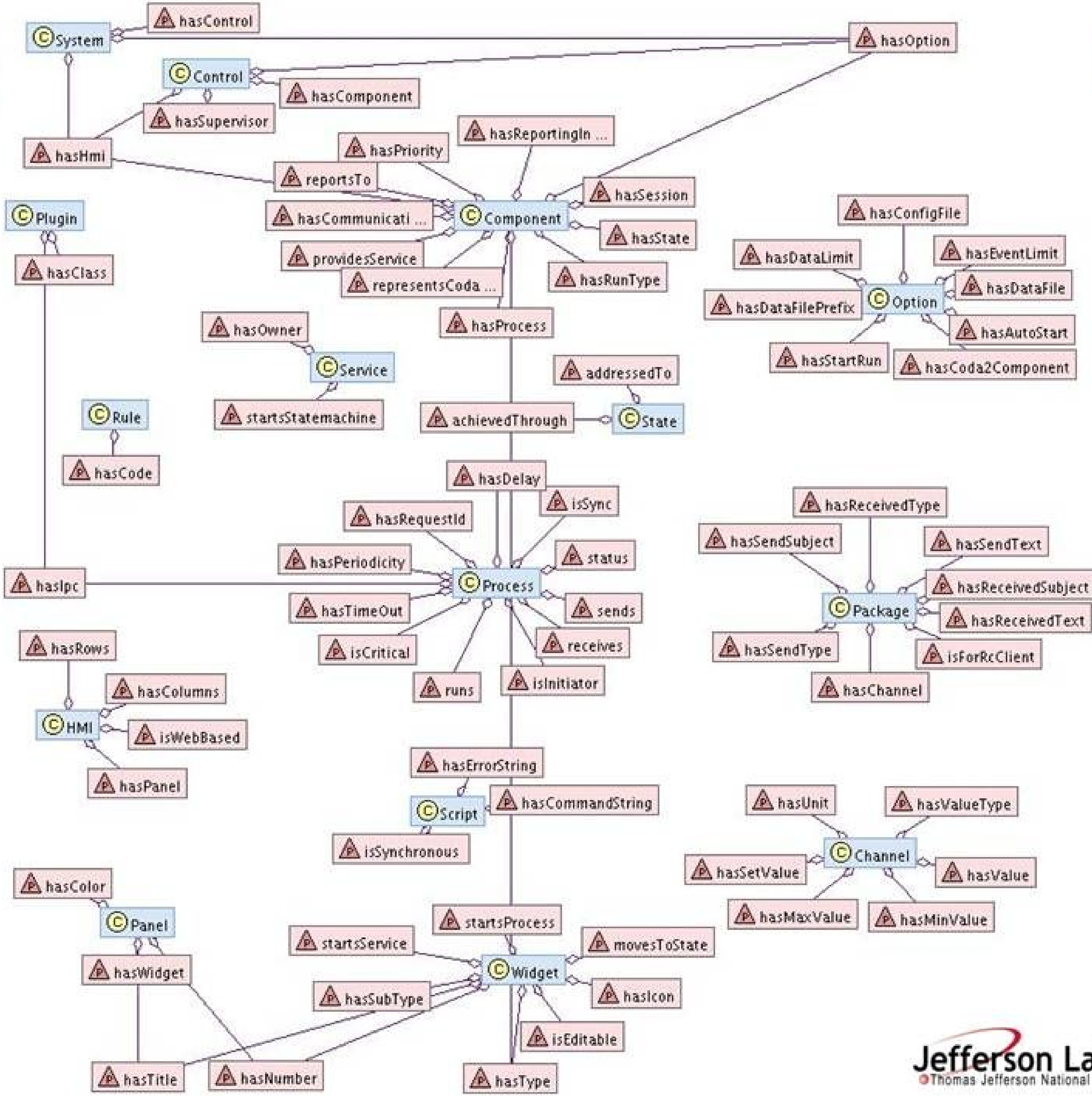
- Language is using RDF as a meta-data modeling specification
- Triplet structure of the language statement
  - Subject – Predicate – Object
  - Concept – Property – Concept/Terminal

Control	*hasComponent*	***HVMainframe*.**
HVMainframe	*hasState*	***HVState1*.**
HVState1	*achievedThrough*	***HVProcess1*.**
HVProcess1	*sends*	***Package1***
Package1	*hasChannel*	***Channel1***
Channel1	*hasSetValue*	Value
HVMainframe	*providesService*	***Service1***
Service1	*hasCode*	CodeDescription

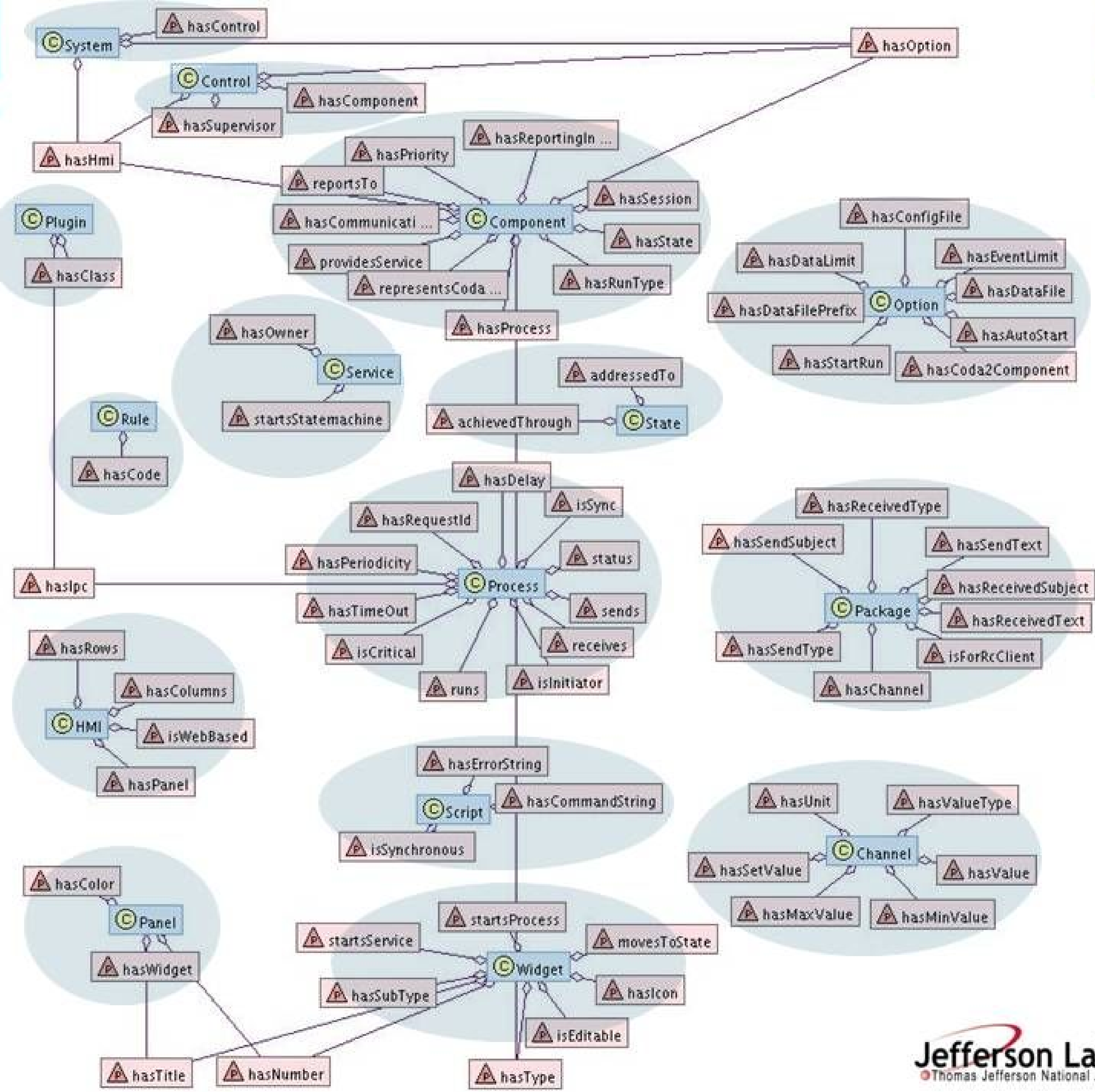
Jefferson Lab
Thomas Jefferson National Accelerator Facility



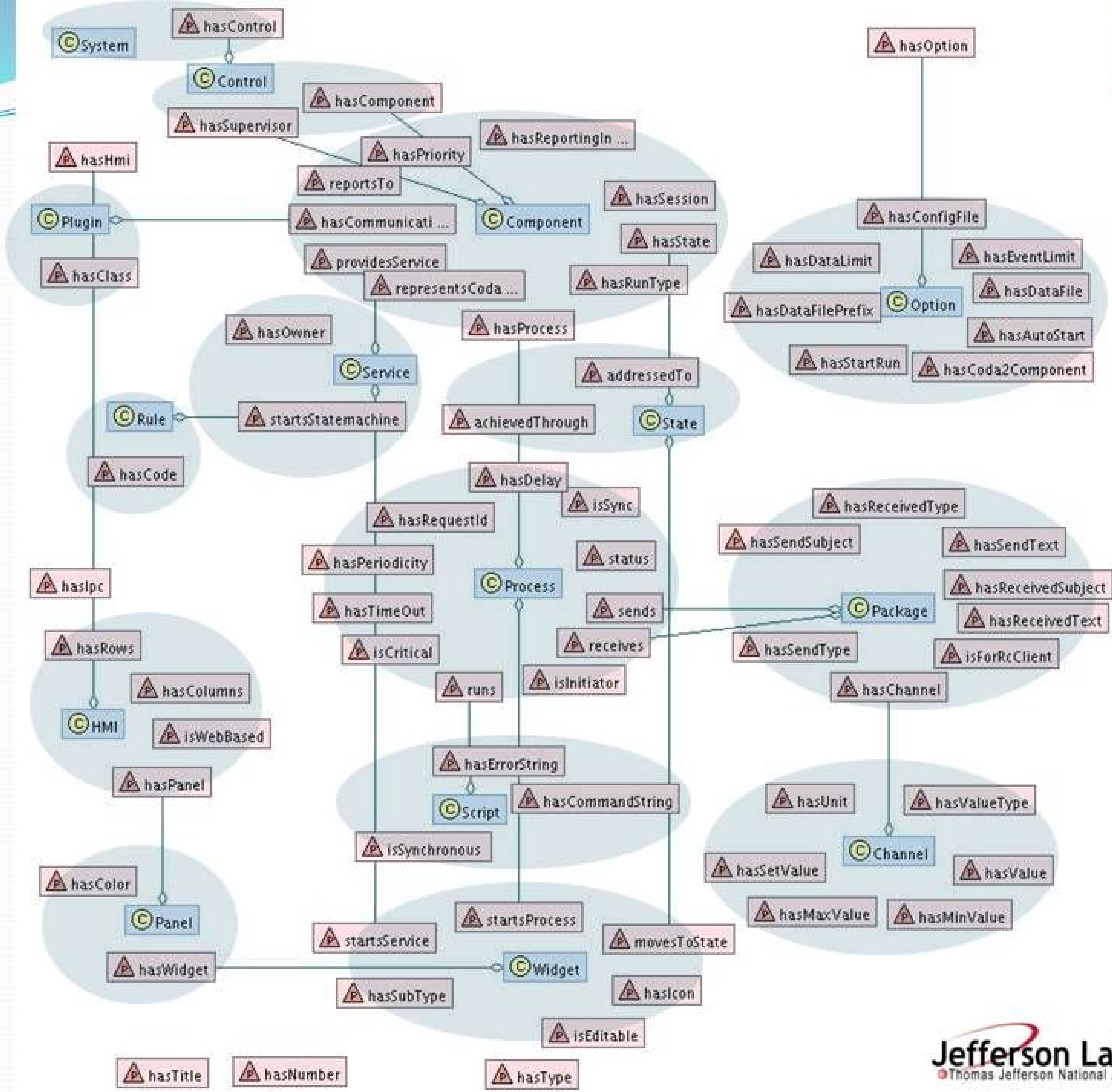
Jefferson Lab
Thomas Jefferson National Accelerator Facility



Jefferson Lab
Thomas Jefferson National Accelerator Facility



Jefferson Lab
Thomas Jefferson National Accelerator Facility


System
hasControl
Control
hasOption
hasComponent
hasSupervisor
hasReportingIn ...
hasPriority
hasHmi
reportsTo
hasSession
hasConfigFile
Plugin
hasCommunicati ...
Component
hasState
hasDataLimit
hasEventLimit
providesService
hasClass
representsCoda ...
hasRunType
hasDataFile
hasDataFilePrefix
Option
hasOwner
hasProcess
hasAutoStart
Service
addressedTo
hasStartRun
hasCoda2Component
Rule
startsStatemachine
achievedThrough
State
hasCode
hasDelay
isSync
hasReceivedType
hasRequestId
hasSendSubject
hasSendText
hasPeriodicity
status
Process
hasReceivedSubject
hasIpc
hasTimeOut
sends
Package
hasReceivedText
hasRows
isCritical
receives
hasSendType
isForRcClient
hasColumns
runs
isInitiator
hasChannel
HMI
isWebBased
hasErrorString
hasPanel
hasCommandString
hasUnit
hasValueType
Script
isSynchronous
Channel
hasSetValue
hasValue
hasColor
Panel
startsProcess
hasMaxValue
hasMinValue
startsService
movesToState
hasWidget
Widget
hasSubType
hasIcon
isEditable
hasTitle
hasNumber
hasType

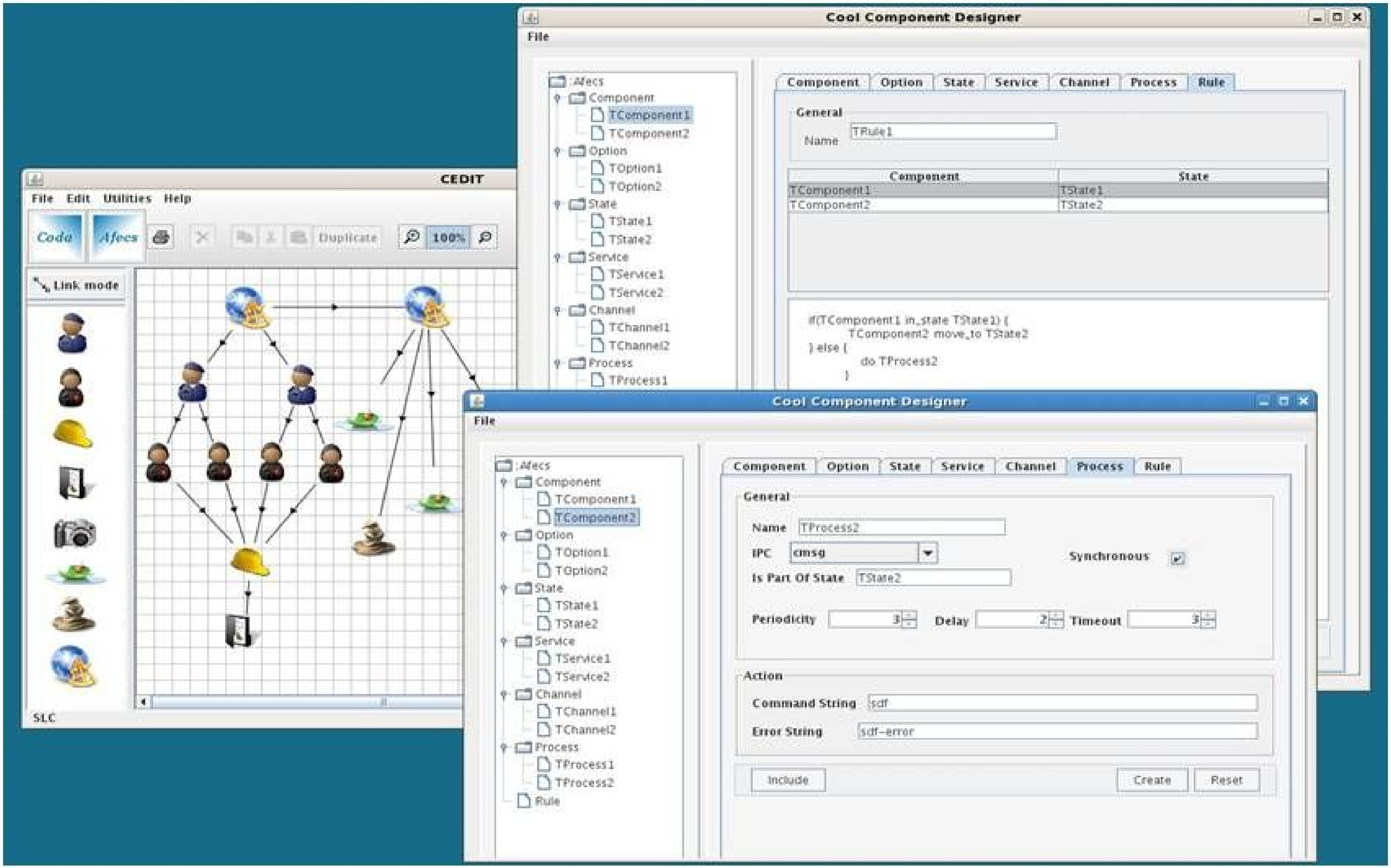
State Machine Description Language

- Terminal node of hasCode action concept is used to describe finite state machine algorithms
- C++/Java like syntax
- && and || logical operators
- Case sensitive

if	elseif	else	do	while
in_state	not_in_state	move_to	true	false
group	supervisor	all	sleep	//

```
If ( ( EB1 in_state EBState1 ) &&
 ( HVMainFrame not_in_state HVState1 ) ) {
    EB1 move_to EBState2;
    do externalProcess1;
} elseif ( HVMainFrame in_state HVState1 ){
   move_to HVState2;
}
```

Graphical Interface



Summary

- Ontology language for describing complex, hierarchical control systems, control logic and finite state machines has been developed
- Language provides following advantages:
  - Increased descriptive power
  - Heterogeneous system component description and integration
  - Graphical interface eliminates direct, programmatic description

Thank you

Cool example coded inXML

```
<rdf:Description rdf:about="http://COOLHOME/Ebs/eb1#EB1">
    <cool:hasIpc>dpsh</cool:hasIpc>
    <cool:representsCoda2Client>true</cool:representsCoda2Client>
    <cool:hasType>EB</cool:hasType>
    <cool:hasName>EB1</cool:hasName>
    <cool:hasCode>{CODA}{CODA}</cool:hasCode>
    <cool:hasPriority>33</cool:hasPriority>
    <cool:hasReportingInterval>3</cool:setsReportingInterval>
<cool:hasState rdf:resource="http://COOLHOME/State/TState#TState1"/>
</rdf:Description>
```