The only thing worse than an experiment you can't control is an experiment you can.

AFECS

Multi-agent Framework for Experiment Control Systems

Vardan Gyurjyan, David Abbott, Graham Heyes,
Ed Jastrzembski, Carl Timmer, Elliott Wolin

Project Goal

- Develop a framework for building control systems
  - open architecture
  - hierarchical
  - network distributed
- Provide a set of tools to deploy:
  - test and control systems
  - alarm systems
  - control visualization systems, etc.

Experiment Control System Requirements

- Control components of the experiment.
- Full description of the experiment components.
- Framework reflects
- Fault tolerance and recovery.
- Create and deploy finite state machines.

Design Philosophy of AFECS

- Pure Java.
- Collaborating autonomous agents.
- Each agent represents a hardware or software component.
- Messages between agents invoke actions.
- Agents behave as finite state machines.
- Agents communicate with physical components through standard protocols.

AFECS Agent Categories

- Normative agents

N

administrative agent
  - Crate and manage all other agents
  - Environment administration

- Supervisor agents

S

Control groups of Component agents

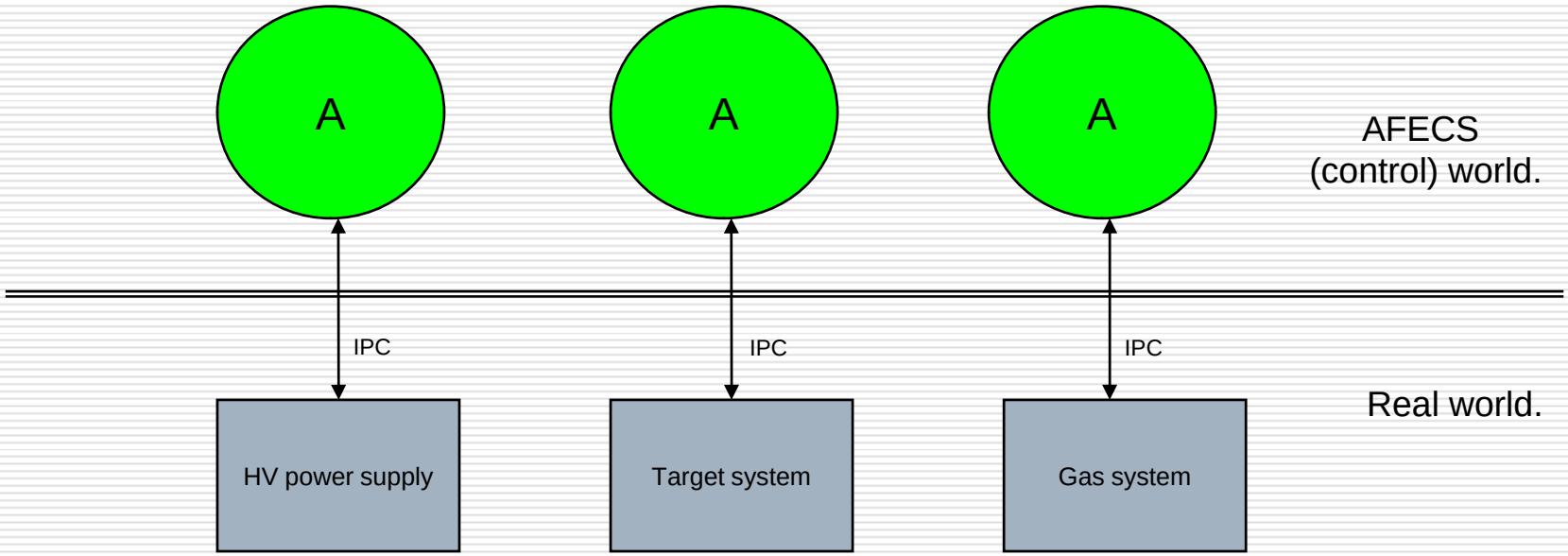
- Component agents

A

Represent hardware or software components of the real world control system.

Component Agent

Agents mimic the state of the real (physical) component, they can invoke actions which change the physical component state.



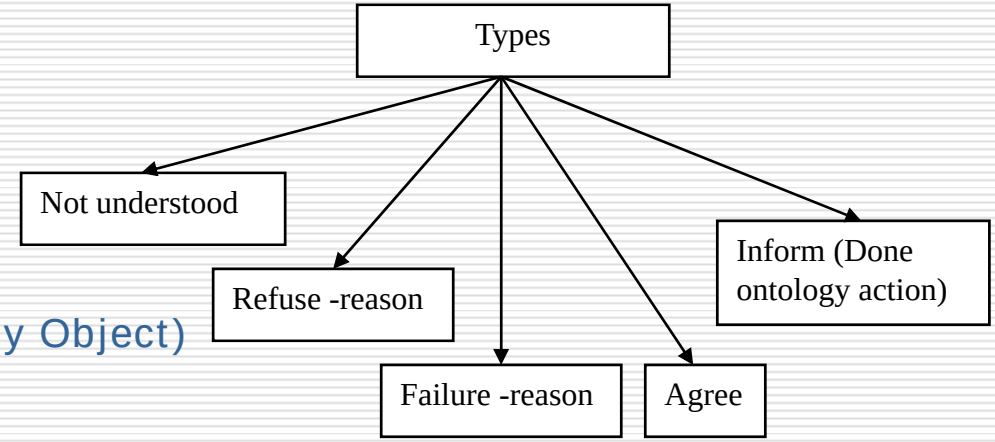
Inter-Process Communication Channel
IPC

Agents communicate with their associated physical components using range of communication protocols including:

- Tcl-DP (legacy protocol)
- cMsg (publish-subscribe messaging protocol, JLAB)
- EPICS channel access protocol
- SNMP (simple network management protocol)
- JDBC
- OS shell interface

Inter-Agent Communication Channel ACC

- All agent communication is through message transfer. Message format is ACL (Agent Communication Language) defined by FIPA (Foundation for Intelligent Physical Agents).
- Each message is one of several predefined types.

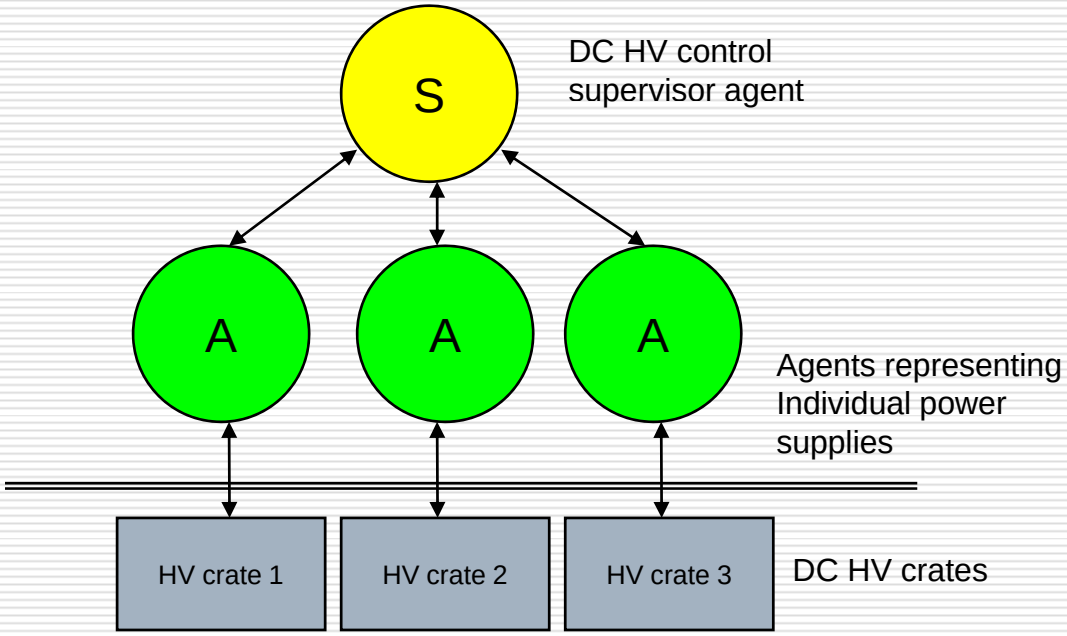


- Message structure:
  - Sender
  - Receiver
  - Content (Data Object or Ontology Object)
  - Language
  - Ontology
  - Protocol

ACC is hidden from the users.

Supervisor Agent

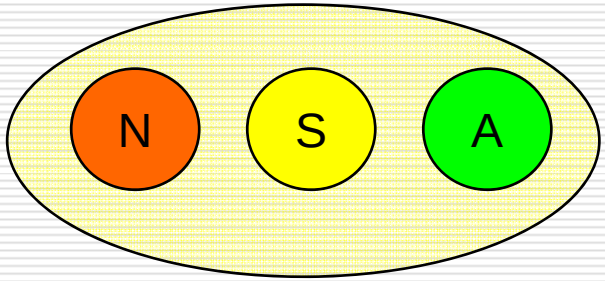
Supervisor agents have discrete states and they respond to messages from other agents (supervisor or component).



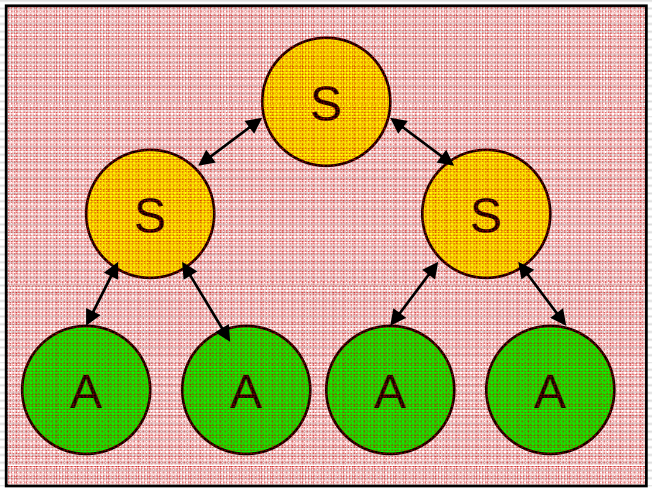
Agents Groupings

- **Physical** (AFECS container)
  A java virtual machine (JVM) is an agent container.

  Container is a process.
  Agent is a set of threads in the process.



- **Logical** (AFECS domain)
  Agents are grouped into virtual clusters or domains according to their functions. Agents in each domain may be visible to other domains.

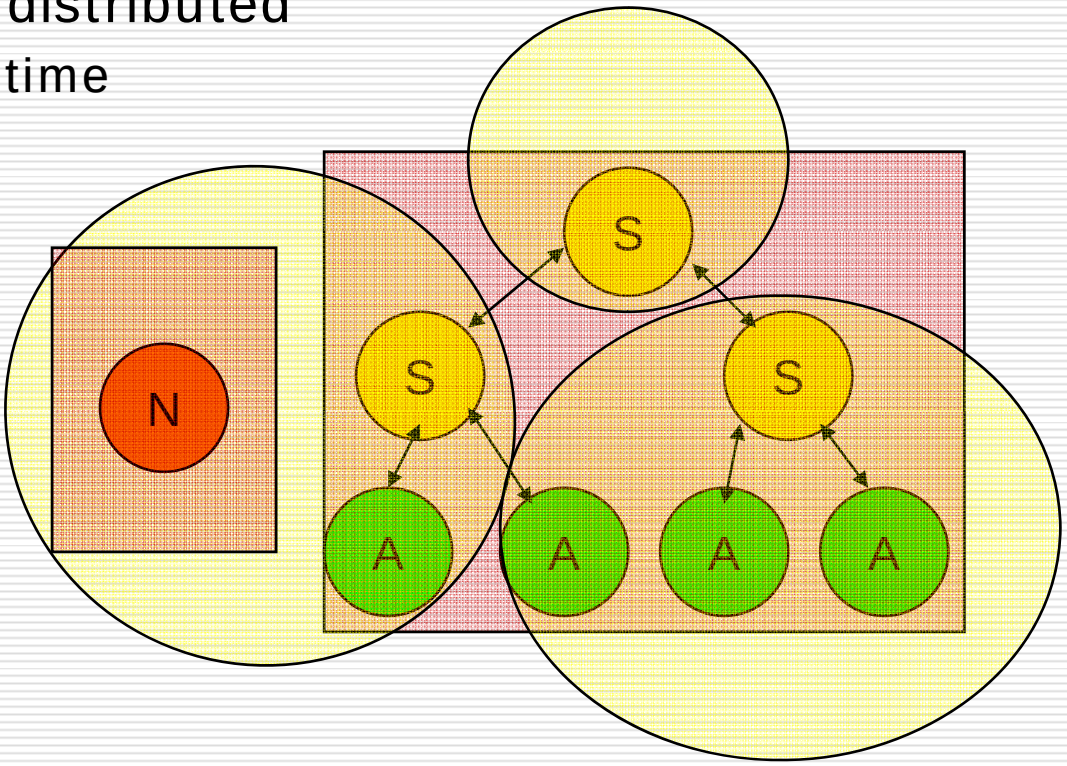


Domain

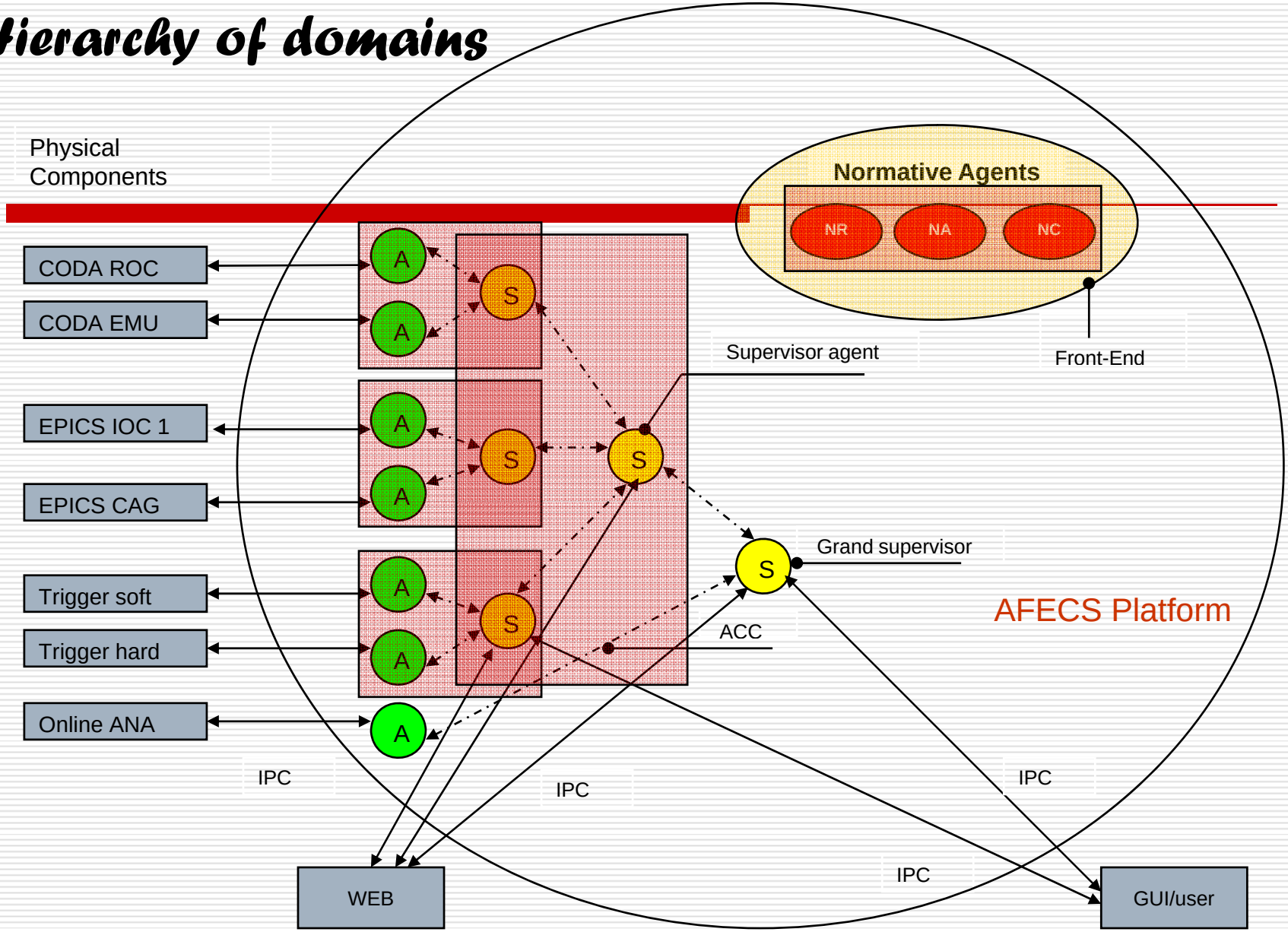
- Reduces complexity of large control systems.
- Encourages modularity and encapsulation.
  (only a few agents and their behaviors are visible outside of the domain limits)
- Promotes control system hierarchical structure.
- Can be redesigned and tested independently, without affecting the rest of the control system.
  (as long as the behavior of the visible agents remains the same)

AFECS Platform Distribution

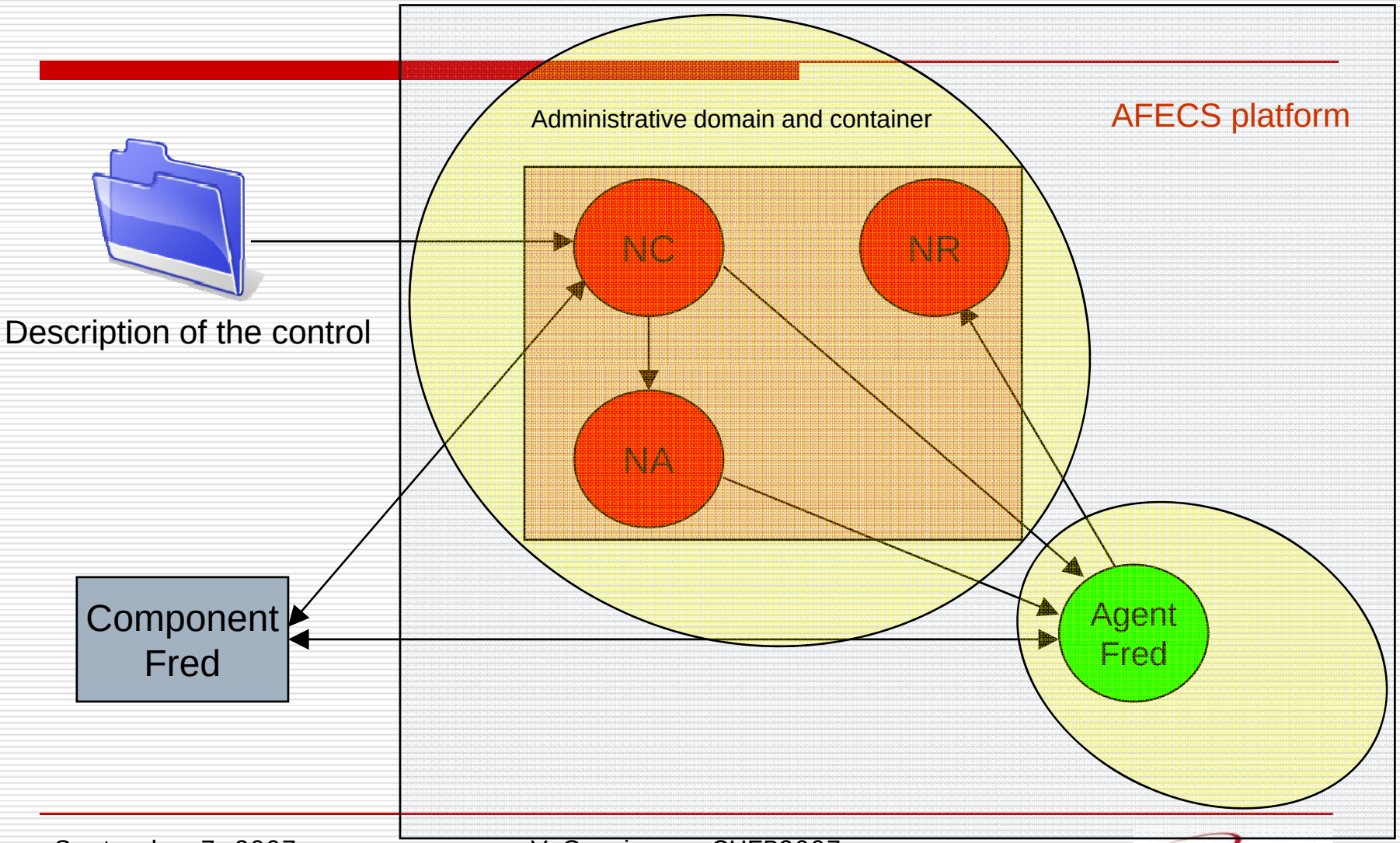
- Agents are physically distributed
  - at the configuration time
  - at runtime



Hierarchy of domains



Physical Component Integration



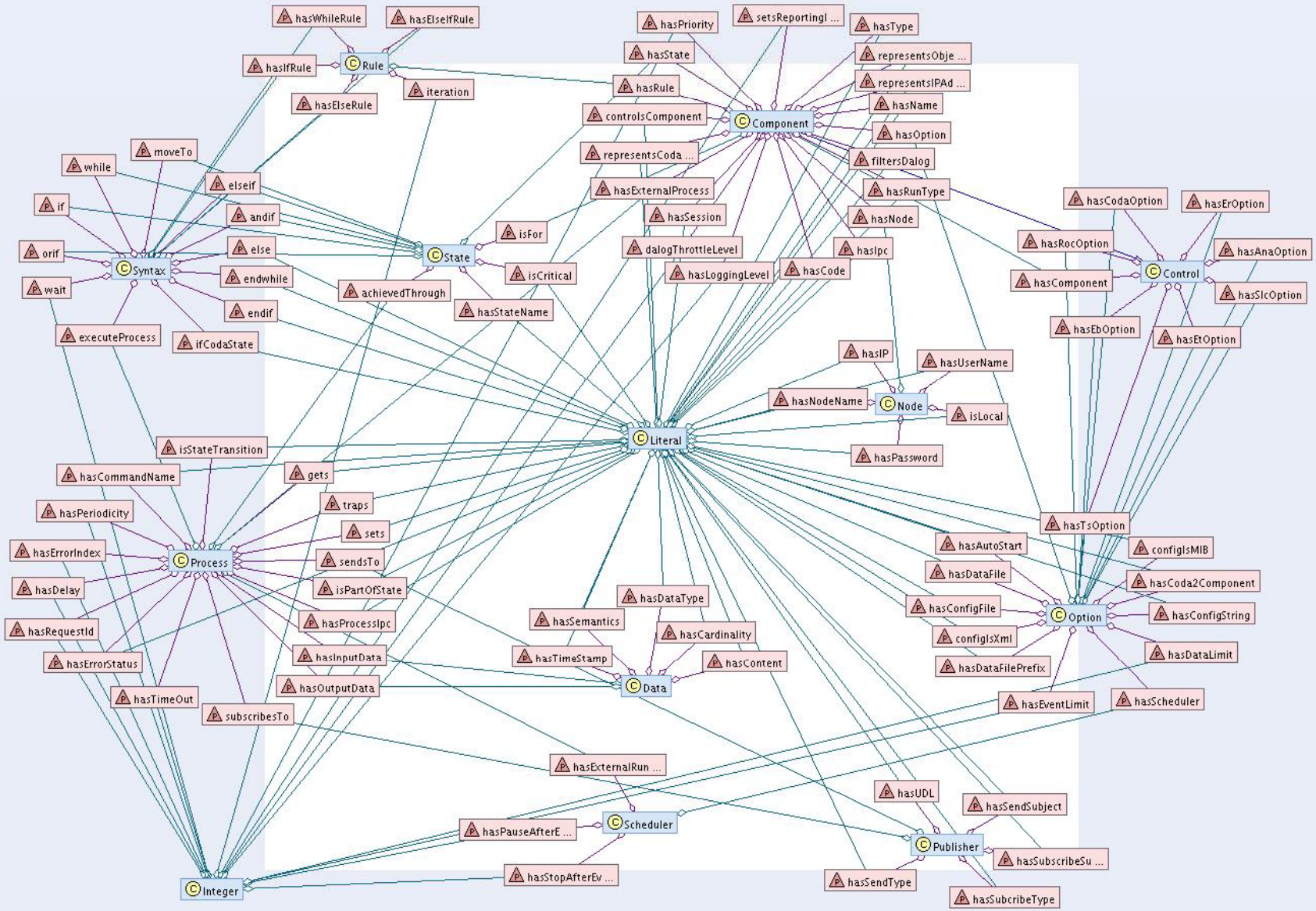
Control System Description

Designing a control system requires precise descriptions of physical components and their software abstractions.

Afecs uses the COOL language for these descriptions.

Control Oriented Ontology Language COOL

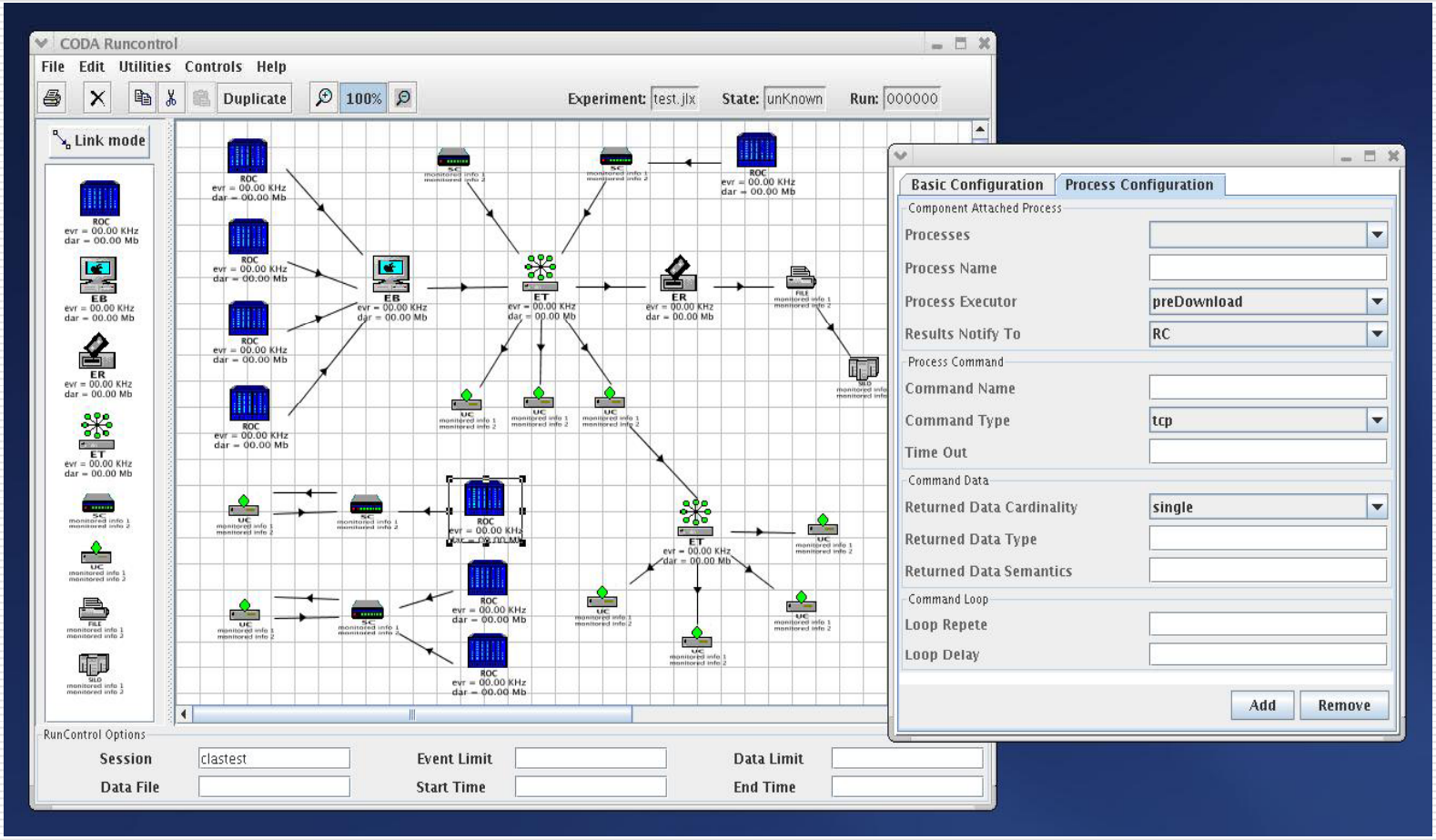
- Defines an agent model of the experiment.
- Describes functionalities of physical components (state machines)
- Describes the semantics of control.
- Developed using RDFS.
- Is intuitive and human readable.
- Can be generated using GUI.


hasWhileRule
hasElseIfRule
hasIfRule
Rule
hasElseRule
iteration
hasPriority
setsReportingI ...
hasType
hasState
representsObje ...
hasRule
representsIPAd ...
controlsComponent
Component
hasName
hasOption
representsCoda ...
filtersDalog
hasExternalProcess
hasRunType
hasSession
hasNode
dalogThrottleLevel
hasIpc
hasLoggingLevel
hasCode
isFor
isCritical
hasStateName
State
achievedThrough
moveTo
while
elseif
if
andif
orif
else
Syntax
endwhile
wait
endif
executeProcess
ifCodaState
hasCodaOption
hasErOption
hasRocOption
hasAnaOption
hasComponent
Control
hasSlcOption
hasEbOption
hasEtOption
hasIP
hasUserName
hasNodeName
Node
isLocal
Literal
hasPassword
isStateTransition
hasCommandName
gets
traps
hasPeriodicity
sets
hasTsOption
hasErrorIndex
Process
sendsTo
hasAutoStart
configIsMIB
hasDataFile
hasDelay
isPartOfState
hasCoda2Component
hasDataType
hasConfigFile
Option
hasConfigString
hasRequestId
hasProcessIpc
hasSemantics
hasCardinality
configIsXml
hasErrorStatus
hasInputData
hasTimeStamp
hasContent
hasDataFilePrefix
hasDataLimit
hasTimeOut
hasOutputData
Data
hasEventLimit
hasScheduler
subscribesTo
hasExternalRun ...
hasUDL
hasSendSubject
hasPauseAfterE ...
Scheduler
Publisher
hasSubscribeSu ...
hasStopAfterEv ...
hasSendType
hasSubcribeType
Integer

AFECS COOL Specific Tools

- GUI using COOL to build configuration files describing control system.
- db2cool: converts current CODA config database to COOL config files.
- sml2cool: translates state manager language config files to COOL config files.

COOL GUI



Jefferson Lab

COOL Example

```
<rdf:RDF'
xmlns:cool='http://coda.jlab.org/COOL/cool#'
>

<cool:Control rdf:ID="R1ExternalProcess">
   <cool:hasComponent rdf:resource="#ROC1"/>
   <cool:hasComponent rdf:resource="#RC"/>
   <cool:hasOption rdf:resource="#OPTION"/>
</cool:Control>
<cool:Option rdf:ID="OPTION">
   <cool:hasCoda2Component>false</cool:hasCoda2Component>
   <cool:hasDataFile>/tmp/test.dat</cool:hasDataFile>
</cool:Option>
<cool:Component rdf:ID="ROC1">
   <cool:hasIpc>cMsg</cool:hasIpc>
   <cool:hasType>ROC</cool:hasType>
   <cool:hasName>Roc1</cool:hasName>
   <cool:hasPriority>44</cool:hasPriority>
   <cool:hasExternalProcess rdf:resource="#PROC1"/>
</cool:Component>
<cool:Component rdf:ID="RC">
   <cool:hasIpc>cMsg</cool:hasIpc>
   <cool:hasType>RCS</cool:hasType>
   <cool:hasName>R1ExternalProcess</cool:hasName>
   <cool:hasPriority>77</cool:hasPriority>
</cool:Component>
<cool:Process rdf:ID="PROC1">
   <cool:hasCommandIpc>shell</cool:hasCommandIpc>
   <cool:isPartOfState>predownload</cool:isPartOfState>
   <cool:hasCommandName>emacs</cool:hasCommandName>
</cool:Process>
</rdf:RDF>
```

AFECS Implementations

- New run control for the JLAB data acquisition system.
  (V. Gyurjyan, et al. “Jefferson Lab Data Acquisition Run Control System”, Proceeding of the CHEP conference. CERN-2005-002, Volume 1, page 151.)

- CLAS experiment web based monitoring system.
  http://clasweb.jlab.org/clasonline/rc/hallB/e-cr.htm

Run-Control GUI

Jefferson Lab

Conclusions

- Java based framework for designing and implementing hierarchical, distributed control systems with intelligent agents.
- Encourages abstraction, encapsulation, and modularity.
- Provides a special ontology language (COOL) to describe hierarchical control structure, control logic, and state machines.
- Has been successfully used to develop run control system for the JLAB data acquisition system, and CLAS experiment web based monitoring system.

Thank you

Terminology

- **Physical Component**. A hardware or software entity, being controlled by the framework. Examples are: CODA ROC, CODA EMU (Event Management Unit, EB, ER, etc.), EPIC IOC, EPICS CAG (channel access gateway), 3rd party control system, 3rd party hardware, Online analyses application, farm node, etc.
- **Agent**. A Java software entity representing actual physical component or components.
- **Supervisor Agent**: Agent encapsulating control domain specific finite state machine, and control specific knowledge base.
- **Normative Agent**: Agent responsible for platform and agent administration and management.
- **Control Domain**. Agents grouped into virtual clusters or domains with their specialized functionalities. For example: DAQ run control domain, DC High voltage domain, Epics domain, etc.
- **Agent Platform**. Distributed environment, containing Main Container and multiple Agent Containers. Provides facilities whereby agents can be located, identified, communicated and manipulated (migration over the containers, cloning, etc.).
- **Agent Container**. Java virtual machine, providing a complete runtime environment for agent deployment and concurrent execution.
- **Main Container or Front-End.** Agent container where normative agents for agent registration and configuration(ARC), agent platform administration (PA), and Jade platform support agents (AMS, DF) are running.
- **IPC.** Inter process communication channel between physical component and representative agent.

	EPICS	SMI++	AFECS
Portability	no	no	yes
Hierarchical control system	no	yes	yes
Web interface	no	no	yes
Control abstraction	no	yes	yes
Standard IPC	yes	yes	yes
Description language	no	yes	yes
Description language extensibility (ontology support)	no	no	yes