Introduction to Brahms

Modeling, Simulation and Development of Multi-Agent Systems with Brahms

Maarten Sierhuis, Ph.D.
Carnegie Mellon University
Silicon Valley
NASA Ames Research Center

Seminar Topics

- Applications 8:30am - 9:30am
- Break 9:30am - 9:45am
- Brahms the Language 9:45am - 10:45 am
- Break 10:45am - 11:00am
- Brahms the Tool (demo) 11:00 - 12:00am

Brahms Tutorial

Application of Human-Centered Work System Design using Brahms

System Centered
The focus is on the work of people in support of the work system.

**What is the delta?**

- How people work together
- How people get to participate in a collaborative activity
- Role of the environment / use of space
- How communication happens
- The tools and artifacts used
- Individual motives, history and culture
- Not just problem-solving activities

**Theoretical**

- Brahms is a agent-based modeling and simulation environment
- Simulation as a work system design/analysis tool
- Agents represent social and collaborative Humans
- Agents are situated, deliberative, cognitive and reactive
- Understanding how people really work
- Developing multi-agent systems

**Practical**

- Mobile Agents: Human-Robot & Human-Agent Interaction
- OCAMS: M&S of OCA Flight Controller in Mission Operations at JSC

**Applications**

Brahms is a agent-based modeling and simulation environment. Simulation is a work system design/analysis tool. Agents represent social and collaborative Humans. Agents are situated, deliberative, cognitive and reactive. Understanding how people really work and developing multi-agent systems.

**BRAHMS – HISTORY OF APPLICATIONS**

- TRL 1: NYNEX
- TRL 2: APOLLO
- TRL 3: JSC
- TRL 4: JPL
- TRL 5: JSC MOD OCAMS PROJECT
- TRL 6: Aeronautics CTFM PROJECT
- TRL 7: Supported by NASA and Scout robot crew in Space 8.5 Test & High Test in 2004
- TRL 8: JSC MOD OCAMS PROJECT
- TRL 9: JSC MOD OCAMS PROJECT
- TRL 10: JSC MOD OCAMS PROJECT
Human-Agent Communication

H => A and A => A

Brahms Communication Architecture
Multi-Agent Simulation to Implementation: A Practical Engineering Methodology for Designing Space Flight Operations

William J. Clancey, Maarten Sierhuis, Chin Seah, Chris Buckley, Fisher Reynolds, Tim Hall, Mike Scott

Outline

- Problem: Earth-Orbit Communications
- Approach: Simulation-to-Implementation
- Model of Current Operations Work System
- Model of Future Operations Work System
- Related Work
- Implications for Software Engineering
Problem: OCA Console Operations

And, oh by the way... log everything you do!!

Solution: OCAMS – OCA Mirroring System

During STS flight #118, files manually transferred:
Uplinked = 2,513 files or 268 MB
Downlinked = 8,411 files or 29.4 GB

Complex Work System
- People & Organizations
- Computer Systems
- Communication Media
- Space Comm Network
- Geographic Distribution
- Regulations
- Work practices & protocols

Approach: Simulation to Implementation
- Current Ops Simulation
- Work System Design
- Future Ops Simulation
- Workflow Tool

Observation
Implementation
Metrics & Data
Operations
NB: Current Ops Model simulates OCA officer & uses one Com. Agent to interface with Excel; delivered March 2007

File types & folder paths are modeled to facilitate later automation.

NB: Future Ops Model runs on one laptop; delivered October 07
Future Ops: Simulation GUI (OCAMS Prototype Tool)

Statistics: Manual (current) vs. Automated (future) OCA Mirroring

Current Operations: Mirroring Activities
\[ > 5\% \text{ shift time} \]

Future Operations (with OCAMS):
Mirroring Activities
\[ < 0.5\% \text{ shift time} \]

Related Work

- Workflow Management Tools
  - Unit of Analysis: Activity vs. Task
  - Run-time Process: Identity vs. Function
  - Simulation fits Implementation-in-Practice

- Agent-Based Modeling & Simulation
  - Cognitive Agents
    - Beliefs, Multiple Groups/Roles
    - Subsumption vs. Procedural Stack
  - Contextual Behavior fits Distributed Work System

Implications for Software Engineering

- Highly interdisciplinary methodology for engineering complex distributed applications
  - Ethnography, Flight Ops (Aeronautics), Computer Science/Networking, Space Science

- Simulation-to-Implementation: Analysis, design, development & verification of agent system

- Agent-Based Systems Integration: Middleware infrastructure for agent societies
**BREAK**

- 15 minutes break!!!

---

**Brahms Tutorial**

The Brahms Language

---

**Types of Agent Languages**

- NetLogo
- Repast
- Swarm
- Swarm-based
- Brahms
- Object Based
- BDI
- Agent Oriented
- Agent Simulation
- Jade
- JADEX
- Jack
- BDI
- Java-based
- Object Based
- Jason
- 3APL
- Brahms
- Subsumption-based
- Reactive Brahms
- Organization Model
- Geography Model
- Agent Oriented Modeling
- Rule Based Cognitive Modeling
- Agent Based Modeling

---

**Overview of Brahms Agent Environment**

Agent Environment consists of the following:

- **Brahms Agent Oriented Language**
- **Composer** for building models.
  (or Integrated Development Environment).
- **Compiler** for compiling models.
- **Virtual Machine** for simulating models.
  (or Simulation Engine).
- **Agent Viewer** for viewing simulations.
**Brahms Language**

- **Agent-Oriented / BDI**
  - Agents are first-class citizens
  - Agents are belief-based
  - Agents are intention-based
  - Beliefs become intentions that trigger reasoning- and/or situation-action rules

- **Organizational Modeling**
  - Agents can be modeled within an hierarchical member-of inheritance structure

- **Object-based**
  - Objects can represent physical artifacts, data and concepts to reason with
  - Integration of Java objects as data objects, Java activities and Java agents

- **Geography-based**
  - Areas can be conceptual representations of locations
  - Areas can be located within other areas, creating a hierarchical environment model
  - Agents and objects can be located within an area

**What Brahms Models Include**

- **Agent Organization**
  - Group membership inheritance hierarchy
- **Artifacts in the world as objects relevant in activities**
  - Tools and artifacts people use
  - Class hierarchies
- **Data as objects**
  - Information modeling
- **Environment and its state**
  - Represented hierarchically as areas with sub-areas
  - Agents and objects are located within areas
  - World State (facts vs. beliefs of agents)
  - Defining facts (reactive)
- **Behavior as situated activities that take time**
  - Constraint on beliefs (workframes)
  - Primitive or composite (decomposed)
- **Goal-directed reasoning behavior as production rules within an activity that takes time**
  - Forward-chaining rules that take no time (thoughtframe)

**Anatomy of a Brahms Model**

<table>
<thead>
<tr>
<th>Language Concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Groups</td>
</tr>
<tr>
<td>Agents</td>
</tr>
<tr>
<td>Classes</td>
</tr>
<tr>
<td>Objects</td>
</tr>
<tr>
<td>AreaDefs</td>
</tr>
<tr>
<td>Areas</td>
</tr>
<tr>
<td>Paths</td>
</tr>
<tr>
<td>Conceptual Classes</td>
</tr>
<tr>
<td>Conceptual Objects</td>
</tr>
<tr>
<td>Attributes (OA-V)</td>
</tr>
<tr>
<td>Relations (ORO)</td>
</tr>
<tr>
<td>Beliefs</td>
</tr>
<tr>
<td>Facts</td>
</tr>
<tr>
<td>Activities</td>
</tr>
<tr>
<td>Workframes</td>
</tr>
<tr>
<td>Preconditions</td>
</tr>
<tr>
<td>Consequences</td>
</tr>
<tr>
<td>Detectables</td>
</tr>
<tr>
<td>Thoughtframes</td>
</tr>
</tbody>
</table>

**Beliefs-Desires-Intentions**

- **Brahms Agent**
  - Beliefs (static, heredity)
  - Plans (production, thoughtframes, intentions)
- **Input**
  - Preconditions
  - Workframes
- **Engine**
  - Thoughtframes
- **Output**
  - Intentions
Brahms: agent-oriented language
Symbolic Discrete Event BDI-like System

Production and Situation-action Rules

WF1: C1 and C5 and C7 => Detect F8, Activity1(t), B5, B7
WF2: C4 => Activity2(t), B6
WF3: C5 and C6 and C8 => Detect F9, Activity1(t), B5, B7

WF1, WF2, WF3...

Belief Memory
B1, B2, B3, ........

Next time event
State at next time event

WF1: C1 and C5 => B2
WF2: C8 => B9
WF3: C9 and C6 and C8 => B10

Thoughtframe Rule Memory

WF1, WF2, WF3...

Agent

World State

Brahms Agent Engine

Brahms Virtual Machine

Brahms: Distributed MAS
Brahms Groups, Agents and Attributes:

What is a Brahms agent?

- Agents model human behavior.
- Agents could be autonomous intelligent systems.
- Attributes of an agent:
  - autonomy,
  - social ability,
  - reactivity,
  - pro-activeness,
  - mobility
  - bounded rationality.

What is a Brahms group?

- A Brahms group describes the abstract properties and behaviors of a group of agents.
- Types of groups:
  - Functional
  - Organizational
  - Social
- Groups can be members of multiple groups.
- Agents can be members of multiple groups.

What are Brahms attributes?

- Attributes represent a property of a group/class or agent/object.
- Attributes have values.
- Currently, only allow single-valued attributes.
  - Recently added a Map collection-type attribute. Maps allow for the assignment of multiple values to the attribute where each value is addressable using an index or key.
- Scope of an attribute:
  - Private – cannot be inherited.
  - Protected – access only for members of group.
  - Public – access by any group or agent.
- Attribute values are assigned or changed by asserting new beliefs or facts.
Brahms Facts & Beliefs:
What is a Brahms belief?

• Represents an agent’s interpretation of a fact in the world.
  – “South Hall is 65 degrees but Alex believes its 80 degrees.”
• Represents an agent’s conception of the world (s)he lives in.
  – “I am a student at University of California, Berkeley.”
• Beliefs are “local” to an agent.
• Agents can reason about their beliefs.
• Agents can communicate their beliefs.

Brahms Facts & Beliefs:
What is a Brahms fact?

• Represent some physical state of the world.
  – Alex is male is true.
• Facts are globally true in the world.
• Agents do not reason with or act directly on facts.
• Agents can detect facts in the world (which represents noticing or sensing).

Beliefs versus Object Attribute values

Object Orientation

```java
object Alex instanceof class student {
    Public Boolean male = true;
}
```

```java
object Kim instanceof class student {
    Public Boolean male = false;
}
```

Belief-based

```java
agent Alex memberof Student {
    public boolean male;
    initial_beliefs:
    (current.male = true);
    (Kim.male = false);
}
```

```java
agent Kim memberof Student {
    initial_beliefs:
    (Alex.male = false);
    (current.male = false);
}
```

Relationship between Brahms Facts and Beliefs
Brahms Facts & Beliefs

World

Fact: (It is 10:40AM)
Fact: (It is Thursday Nov. 19, 1969)
Fact: (the door of the SEQBay is closed)
Fact: (AlBean is located in the SEQBayArea)
Fact: (PeteConrad is located in the SEQBayArea)
Fact: (PeteConrad is ready to offload the ALSEP)
Fact: (PeteConrad is located in the SEQBayArea)

Agent

belief: (the door of the SEQBay is closed)
belief: (AlBean is located in the SEQBayArea)
belief: (PeteConrad is located in the SEQBayArea)
belief: (PeteConrad is ready to offload the ALSEP)

Brahms Thoughtframes

• When student is studying
  – Assess How Hungry you are
  – Based on norms about how much you want to spend and how hungry you are, determine how much cash you need
  – Based on cash available and costs of meals at

How long before I am hungry?

• IF an hour has gone by THEN my hunger level goes up by 3

thoughtframe thoughtframe-name {
  display: literal-string;
  repeat: truth-value;
  priority: unsigned;
  variables: variable declaration
  when(precondition-declaration)
  do {
    thoughtframe-body-element
  }
}

thoughtframe-name ::= name
thoughtframe-body-decl ::= do[ thoughtframe-body-element ; ]* thoughtframe-body-element ::= CON.consequence
**Need Cash To Eat?**

- IF amount of cash < $12 and preferred cash in pocket
  THEN need cash
- IF amount of cash >= $12 THEN don’t need cash

**Which Diner?**

- IF amount of cash > $15 THEN can afford Blakes diner.
- IF amount of cash <= $15 THEN cannot afford Blakes so choose Raleigh

**Forward Reasoning**

**Activities**

- socially constructed engagements,
- situated in the real world,
- taking time, effort and application of knowledge,
- defined beginning and end,
- NOT necessarily need goals in the sense of problem-solving tasks,
- can be interrupted.

When in an activity people might articulate a task they are working on, and the goal they want to accomplish (Clancey '97)

Tasks and Goals are constructed within an Activity (Clancey '97) (Kuutti '96)
What are some of Alex’s activities?

- Studying in South Hall.
- Walking to a restaurant.
- Getting money from a bank’s cash machine.
- Eating food at diner.

Characterization:
- has a duration
- is situated in the real world
- can be interrupted / resumed, but stay active
- can be decomposed and/or subsumed

Brahms Activities

- Primitive activities
  - Lowest level, user-defined, but not further specified.
  - Parameters are time, and resources
- Predefined activities
  - Primitive activities with predefined semantics (communicate, move, etc.)
- Composite activities
  - User-defined detailed activities
  - Decomposed in sub-activities
  - Describes what an agent does while “in” the activity
- Java activities
  - User-defined primitive activities that are implemented in a Java class
  - Uses the Brahms API.

Primitive Activities

```plaintext
primitive-activity ::= primitive_activity activity-name
{ (param-decl [param-decl ?]) }
{ (display [ID.literal-string])
  (priority [ID.unsigned | param-name])
  (random [ID.truth-value | param-name])
  (min_duration [ID.unsigned | param-name])
  (max_duration [ID.unsigned | param-name])
  resources : [param-name | OBJ.object-name]
  [ and [param-name | OBJ.object-name] ]*
}

primitive_activity Study [Books course_book]
{ display : "Study for a Course"
  (priority : 10)
  (random : false)
  (min_duration : 1800 /* 30 mins */)
  (max_duration : 7200 /* 2 hours */)
  resources : course_book
}
```

Predefined Primitive Activities (1)

- Create Agent/Object/Area
  - Creates new agents/objects/areas dynamically
    - Agent can be member of multiple groups
    - Objects can be an instance of a class
    - Can bind new agent/object to a return parameter
    - Can give new agent/object a name and location
- Move
  - Moves an agent/object from one area to another area.
    - Agent/object is moved
    - Contained objects/agents are moved
    - Location beliefs/facts are retracted/created
    - Location facts are created
Predefined Primitive Activities (2)

• Communicate
  – Communicates agent’s beliefs from/to receiver agent(s)
    • Agent needs to have the beliefs to communicate
    • Can specify type of communication (phone | fax | email | face2face | terminal | pager | none)
    • Can specify which agents/objects is being communicate
    • Can specify when to communicate (start | end)

• Broadcast
  – Communicates agent’s beliefs from/to all agent(s) in specific areas
    • Can specify areas and subareas
    • Agent needs to have the beliefs to communicate
    • Can specify type of communication (phone | fax | email | face2face | terminal | pager | none)
    • Can specify to which agents/objects is being communicated
    • Can specify when to communicate (start | end)

Predefined Primitive Activities (3)

• Get
  – Allows an agent or object to pick up or transfer one or more objects and/or agents from an area, other agent or object, and carry it with it while performing activities.

• Put
  – Allows an agent or object to put down (drop) or transfer one or more objects and/or agents, referred to as items, carried by the agent or object performing the activity.

• Gesture
  – The gestures as indicated by the gesture activity are visualized in the virtual reality environment provided that environment supports the specified gestures.

Move Activity

```move moveToRestaurant() {
  location: Telegraph_Av_2405;
}
move moveToSouthHall() {
  location: SouthHall;
}
move moveToLocationForCash(Building loc) {
  location: loc;
}
move moveToLocation(Building loc) {
  location: loc;
}
```
Brahms Workframes

- Workframes are situation-action rules:
  - Activities are associated with a conditional statement or constraint.
  - Workframes are different from production rules, in that they take time.
  - If the conditions of a rule are believed, then the associated activities are performed.
- We call these preconditions
- Workframes can be associated with groups/agents and classes/object.
  - A workframe defines when an activity (or activities) of an agent/object may be performed.
- Having two or more agents with different workframes, performing the same activity, can represent individual differences.
- Conclusions are facts or beliefs or both that may be asserted when a workframe is executed.

**Workframe Syntax**

```
workframe workframe-name
  
  activities: primitive_activity eat( )  
  priority: 0; 
  max_duration: 400;  
  
  workframe wf_eat { 
    repeat: true; 
    variables:
      forone(Cash) cs;
      forone(Diner) dn; 
      when(knownval(current hasCash cs) and 
          knownval(current.location = dn.location))
      do { 
        eat(); 
        conclude((current.howHungry = current.howHungry - 3.00), bc:100, fc:0);  
        conclude((cs.amount = cs.amount - dn.foodcost), bc:100, fc:100);  
        conclude((current.readyToLeaveRestaurant = true), bc:100, fc:0);  
      }
  }
```

Brahms Detectables (for reactive behavior)

- Associated with workframes and activities
- Active while a workframe/activity is active
- Used for:
  - Agents noticing states of the world, and being able to act upon those
  - 3-steps: (i) detect fact, (ii) notice (fact becomes belief), (iii) conditionally act on belief
  - Example: do act A until you notice fact F
- Type: continue | impasse | abort | complete | end_activity

**Detectable Syntax**

```
detectable detectable-name { 
  when { whenever | ID.unsinged }  
  detect [ ] 
  then continue | impasse | abort | complete | end_activity'; 
}
```
Composite Activities

- Decompose activities into sub-activities and the workframes that can execute them.
- Defines a workframe-activity hierarchy
- Execution is different than traditional rule hierarchies:
  - Subsumption hierarchy
  - While “in” an activity the higher-level activity is still active.

Composite Activity Syntax

```
composite_activity study() {
    activities:
        primitive_activity reading() {
            max_duration: 1500;
        }
        workframes:
            workframe wf_readingWhileStudying {
                do {
                    reading();
                }
            }
        thoughtframes:
            thoughtframe tf_needCashToEat {
                when (knownval(cs.amount < current.preferredCashOut))
                do {
                    conclude((current.needCash = true));
                }
            }
    ...
}
```

Workframe-Activity Hierarchy

```
Workframe WF
    Activity A_1 (primitive)  
    Activity A_2 (composite)

Workframe WF_1
    Activity A_{1,1} (primitive)  
    Activity A_{1,2} (composite)

Workframe WF_2
    Activity A_{2,1} (primitive)  
    Activity A_{2,2} (composite)
```

Activity Subsumption

```
ACTIVITY 1
  (other activities)
  WF 1

ACTIVITY 2
  (other activities)
  WF 2

ACTIVITY 3
  (other activities)

ACTIVITY 4
  (other workframes)
  WF 4 (impasse)

ACTIVITY 5
  (other activities)

ACTIVITY 6
  (other activities)
```

Instantiation
Workframe/Activity States

Brahms Java Activities

A Brahms Java Activity is a primitive activity but its actual behavior is specified in Java code.
- Java code may cause an action to happen outside the Brahms model completely (e.g. pop-up a dialog that says “hello world”)
- Java code can generate the output values and assign them to unbound variables in Brahms
- Java code can generate new model objects within the Brahms model
- Java code can generate beliefs/facts into objects/agents/areas within the Brahms model
- Java code can integrate Brahms beliefs to external systems

“Hello World” Java Activity
Used in a Brahms Workframe

```java
java sayHelloWorld() {
    max_duration: 0;
    class: "gov.nasa.arc.brahms.atm.extern.HelloWorld";
    when: start;
} // sayHelloWorld

workframe wf_hello {
    repeat: false;
    when() do {
        sayHelloWorld();
    }//do
} //wf_hello
```

Brahms - FACET Integration

“Agentify” FACET
- Wrap FACET as a Brahms Agent
- Design Agent
- Communicative Act
- Create Java Interface to FACET
A Brahms External Agent is a Brahms agent but its actual behavior is specified in Java code. This means:
- Defined in your Brahms program as an external agent with a name
- Java code implements the behavior of the agent
- Java code to communicate (receive/send beliefs) from other Brahms agents
- Java code can access Brahms model
- Java code can generate facts for objects/agents/areas within the Brahms model
- Java code can "agentify" external systems

Brahms Libraries

- Libraries are Brahms Groups/Classes with domain-independent common capabilities
- Brahms comes with a number of libraries
  - Also, we’re always extending Brahms capabilities...
  - Libraries are a useful way to add capabilities without the need to add to the language
- Most libraries provide generic Activities developed as Java activities
- User can create their own libraries
  - Add Libraries to ../AgentEnvironment/Models/brahms
  - Use import statement, just like with Java libraries
- Current Libraries:
  - Calendar
  - Time and Date library
  - Communication
    - FIPA compliant Communicative Acts Library for agent communication
  - Input/Output
    - File IO library for copying/deleting files
    - Java Utilities
    - Utilities for handling Java objects for Brahms Agents and Objects
  - System Utilities
    - Some simple Java print activities and string manipulation

• 15 minutes break!!!
Overview of Brahms

Brahms consists of the following:

- **Brahms Agent Oriented Language**
- **Composer** for building models.
  (or Integrated Development Environment).
- **Compiler** for compiling models.
- **Virtual Machine** for simulating models.
  (or Simulation Engine).
- **Agent Viewer** for viewing simulations.

Where to download Brahms?

Brahms development environment is bundled into an application called the “Brahms Agent Environment”.

- Go to [http://www.agentisolutions.com/download/index.htm](http://www.agentisolutions.com/download/index.htm)

- Requires MySQL 4.x. database to be installed:
  - Go to [http://www.mysql.com](http://www.mysql.com)
  - Note: Read AgentViewer-Readme.html to set the appropriate privileges for the anonymous database user.

- Requires a Brahms License file.
Files created by the Brahms installation are in:

C:\Program Files\Brahms\AgentEnvironment
Brahms Tutorial

Import Brahms Source into Composer.

Brahms Composer

Beliefs and Facts
Workframes
Thoughtframes
Groups and Agents
Classes and Objects
Geography

Tutorial Scenario Animated

What is my PIN?
Amount of Money?
Banking Institution

The Composer

The Composer
Workframe Interruption

group PrimitiveActivityPerformer {
    attributes:
    public boolean execute_PAC_1;
    activities:
    primitive_activity PAC_1(int pri) {
        display: "PAC 1";
        priority: pri;
        max_duration: 900;
    }
    primitive_activity PAC_2(int pri, int dur) {
        display: "PAC 2";
        priority: pri;
        max_duration: dur;
    }
    workframes:
    workframe wf_PAC_1 {
        repeat: true;
        when (knownval(current.execute_PAC_1 = true))
        do {
            PAC_1(1);
            conclude((current.execute_PAC_1 = false));
        }
    }
    workframe wf_PAC_2 {
        repeat: true;
        do {
            PAC_2(0, 1800);
            conclude((current.execute_PAC_1 = true), bc:25);
            PAC_2(0, 600);
        }
    }
}

Printing beliefs

group Printing beliefs
    assigned SystemGroup {
        attributes:
        public boolean execute_PAC_1;
        public int nr_pac_2;
        public int nr_pac_1;
        public double pac_1_2_ratio;
        initial_beliefs:
        (current.nr_pac_1 = 0);
        (current.nr_pac_2 = 0);
        workframes:
        workframe wf_PAC_1 {
            repeat: true;
            variables:
            forone(int) i;
            forone(int) y;
            forone(int) i_plus_one;
            when (knownval(current.execute_PAC_1 = true) and
                   knownval(current.nr_pac_1 = i) and
                   knownval(current.nr_pac_2 = y) and
                   knownval(i_plus_one = i + 1))
            do {
                PAC_1(1);
                conclude((current.nr_pac_1 = i + 1));
                printBelief(current, nr_pac_1, attribute);
                conclude((current.pac_1_2_ratio = i_plus_one / y));
                printBelief(current, pac_1_2_ratio, attribute);
                conclude((current.execute_PAC_1 = false));
            }
        }
        workframe wf_PAC_2 {
            repeat: true;
            do {
                PAC_2(0, 1800);
                conclude((current.nr_pac_2 = current.nr_pac_2 + 1));
                printBelief(current, nr_pac_2, attribute);
                conclude((current.execute_PAC_1 = true), bc:25);
                PAC_2(0, 600);
            }
        }
    }

INFO : Starting virtual machine
INFO : Starting scheduler
INFO : Starting engine for 'Prim_Agt'
INFO : Virtual machine started...
INFO : 1800: belief(Prim_Agt.nr_pac_2 = 1)
INFO : 4200: belief(Prim_Agt.nr_pac_2 = 2)
INFO : 5100: belief(Prim_Agt.nr_pac_1 = 1)
INFO : 5100: belief(Prim_Agt.pac_1_2_ratio = 0.5)
INFO : 7500: belief(Prim_Agt.nr_pac_2 = 3)
INFO : 9900: belief(Prim_Agt.nr_pac_2 = 4)
INFO : 12300: belief(Prim_Agt.nr_pac_2 = 5)
INFO : 13200: belief(Prim_Agt.nr_pac_1 = 2)
INFO : 13200: belief(Prim_Agt.pac_1_2_ratio = 0.4)
INFO : 15600: belief(Prim_Agt.nr_pac_2 = 6)
INFO : 18000: belief(Prim_Agt.nr_pac_2 = 7)
INFO : 20400: belief(Prim_Agt.nr_pac_2 = 8)
INFO : 21300: belief(Prim_Agt.nr_pac_1 = 3)
INFO : 21300: belief(Prim_Agt.pac_1_2_ratio = 0.375)
INFO : 23700: belief(Prim_Agt.nr_pac_2 = 9)
INFO : 26100: belief(Prim_Agt.nr_pac_2 = 10)
INFO : 28500: belief(Prim_Agt.nr_pac_2 = 11)
INFO : 30900: belief(Prim_Agt.nr_pac_2 = 12)
INFO : 33300: belief(Prim_Agt.nr_pac_2 = 13)
INFO : 34200: belief(Prim_Agt.nr_pac_1 = 4)
INFO : 34200: belief(Prim_Agt.pac_1_2_ratio = 0.3076923076923077)
INFO : 36600: belief(Prim_Agt.nr_pac_2 = 14)
INFO : 39000: belief(Prim_Agt.nr_pac_2 = 15)
INFO : 41400: belief(Prim_Agt.nr_pac_2 = 16)
INFO : 43800: belief(Prim_Agt.nr_pac_1 = 3)
INFO : 43800: belief(Prim_Agt.pac_1_2_ratio = 0.375)
INFO : 46200: belief(Prim_Agt.nr_pac_2 = 18)
INFO : 47100: belief(Prim_Agt.nr_pac_1 = 5)
INFO : 47100: belief(Prim_Agt.pac_1_2_ratio = 0.2777777777777778)
INFO : 49500: belief(Prim_Agt.nr_pac_2 = 19)
INFO : 51900: belief(Prim_Agt.nr_pac_2 = 20)
INFO : 52800: belief(Prim_Agt.nr_pac_1 = 6)
INFO : 52800: belief(Prim_Agt.pac_1_2_ratio = 0.3)
Composite Activities

Workframe Activity Hierarchy

\[
T_{n+1} = T_n + 1800 \\
T_{n+2} = T_{n+1} + 900
\]

\[
wf_Being_Alive() \\
wf_In_Coma()
\]

More Information ...

- E-mail: Maarten.Sierhuis-1@nasa.gov
- Brahms URL: http://www.agentisolutions.com