A Arquitetura Cognitiva LIDA

Ricardo R. Gudwin

DCA-FEEC-UNICAMP

gudwin@unicamp.br

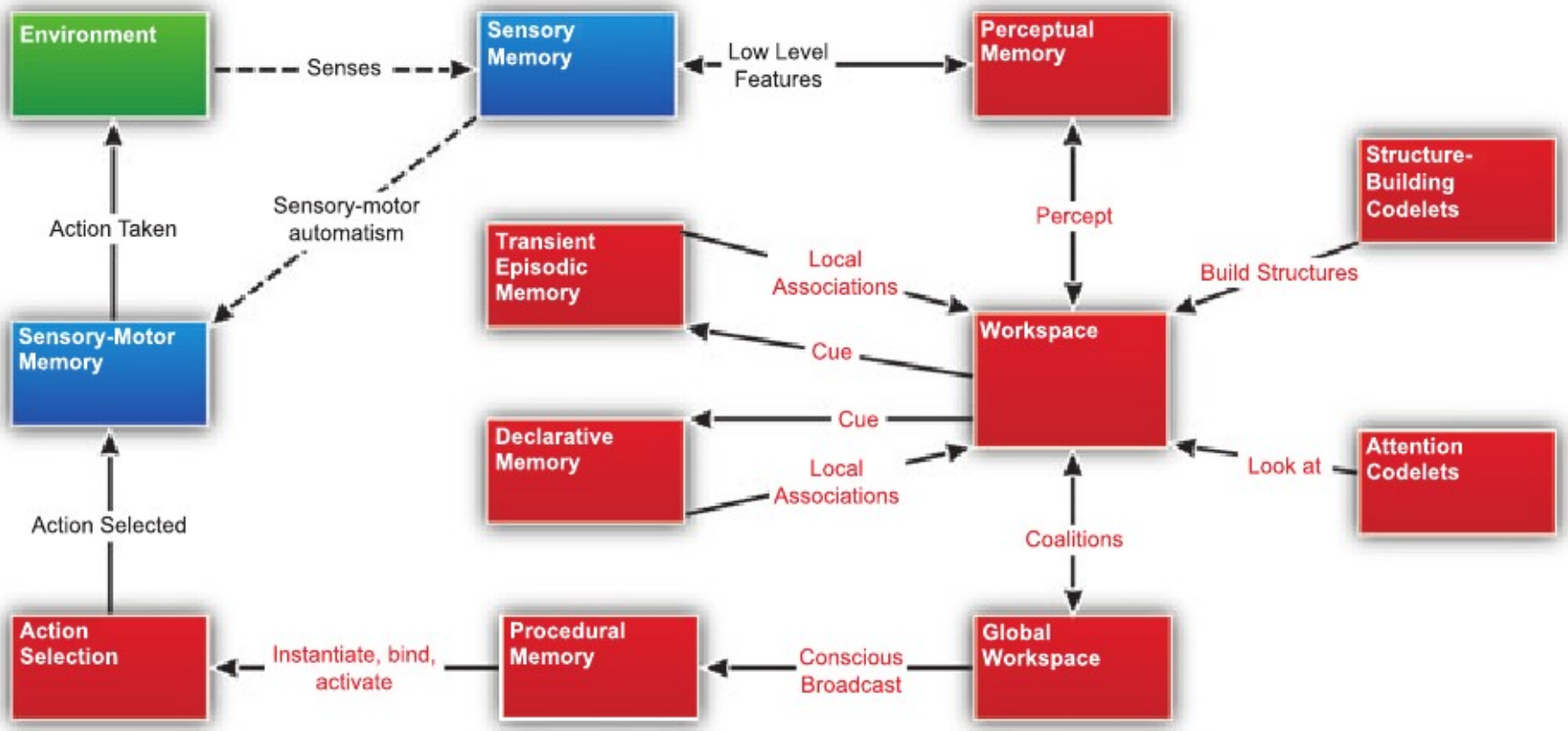
http://www.dca.fee.unicamp.br/~gudwin

A Arquitetura Cognitiva LIDA

- Stan Franklin (Artificial Minds)
  - CCRG: Cognitive Computing Research Group
    - Universidade de Memphis - EUA
  - Conscious Mattie (CMattie – CM - 1999)
    - Baseada na Teoria de um Workspace Global de Baars
  - IDA – Intelligent Distribution Agent (2002)
    - Suporta o modelo de consciência (framework) de Crick & Koch (2003)
  - LIDA – Learning IDA (2006)
    - Novos modelos de aprendizagem
    - Situação em 2012:
      - Versão 1.2 disponível para download: Framework Java
      - http://ccrg.cs.memphis.edu/framework.html

GROUP FOR RESEARCH ON ARTIFICIAL COGNITION

A Arquitetura Cognitiva LIDA



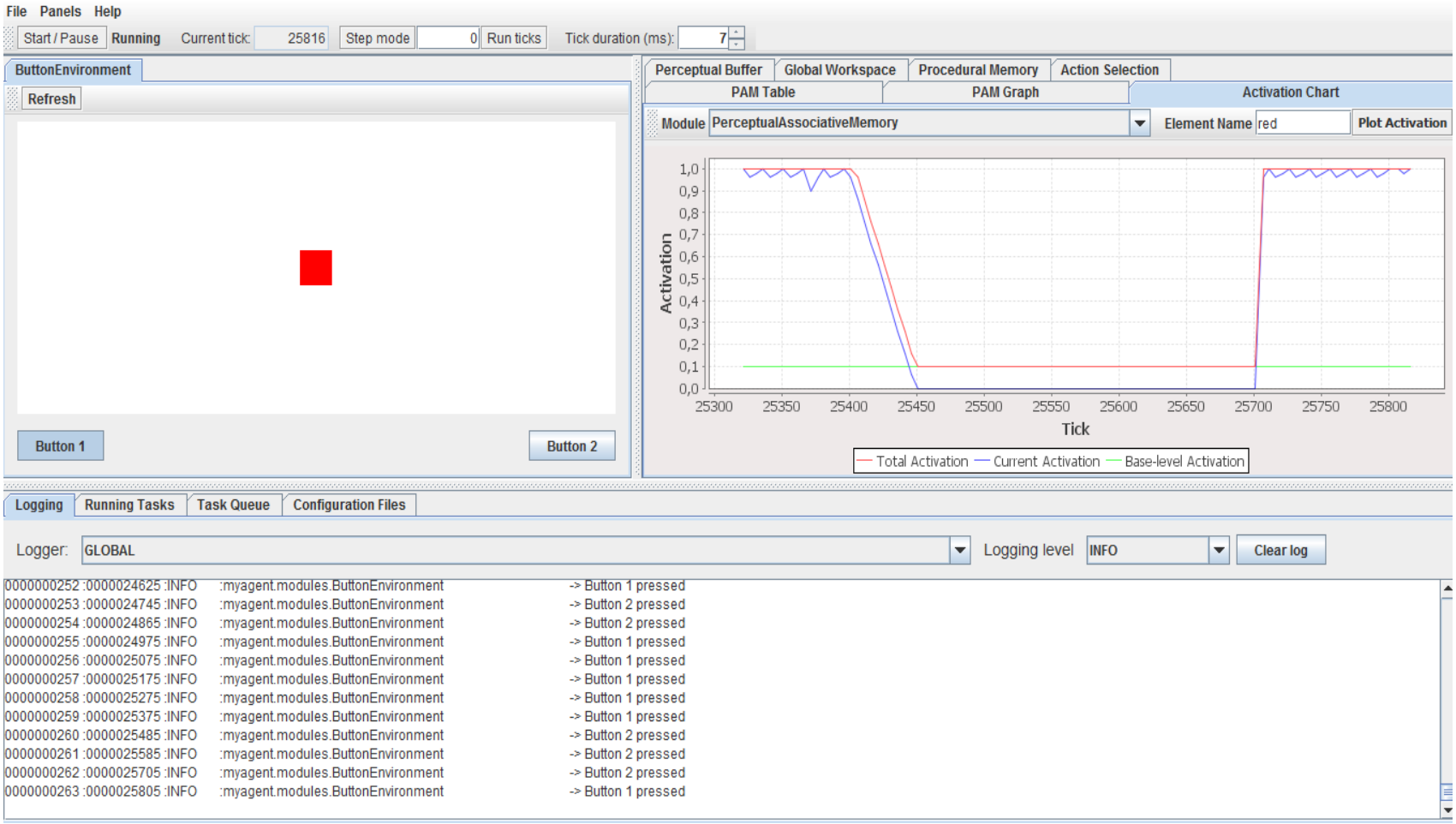
UNICAMP

LIDA Framework

- É uma library Java que constitui um “esqueleto” para o desenvolvimento de sistemas cognitivos baseado no Modelo LIDA.
- Para a criação de agentes com comportamentos inteligentes, é necessário a implementação de interfaces e classes específicas do framework e algumas configurações devem ser feitas através de arquivos XML.
- Possui uma GUI (Graphical User Interface) que permite uma visualização em tempo real do conteúdo de cada módulo, valores associados a parâmetros, tasks e valores associados a variáveis.

GROUP FOR RESEARCH ON ARTIFICIAL COGNITION

LIDA GUI



UNICAMP

LIDA Framework

- Conceitos Gerais
  - O Framework é composto por diversos **Modules**, de diversos tipos diferentes
  - Para muitos **Modules**, existem diferentes implementações já providas pelo Framework. Deve-se apenas escolher se o **Module** irá ou não ser utilizado
  - Alguns Modules são específicos da aplicação, e devem ser gerados pelo usuário do Framework
  - Modules são implementados a partir da execução cíclica de **FrameworkTasks**, que são gerenciados a partir de um **TaskManager**
  - Cada Module pode ter seu **TaskSpawner**, que executará seus FrameworkTasks em diferentes frequências de execução

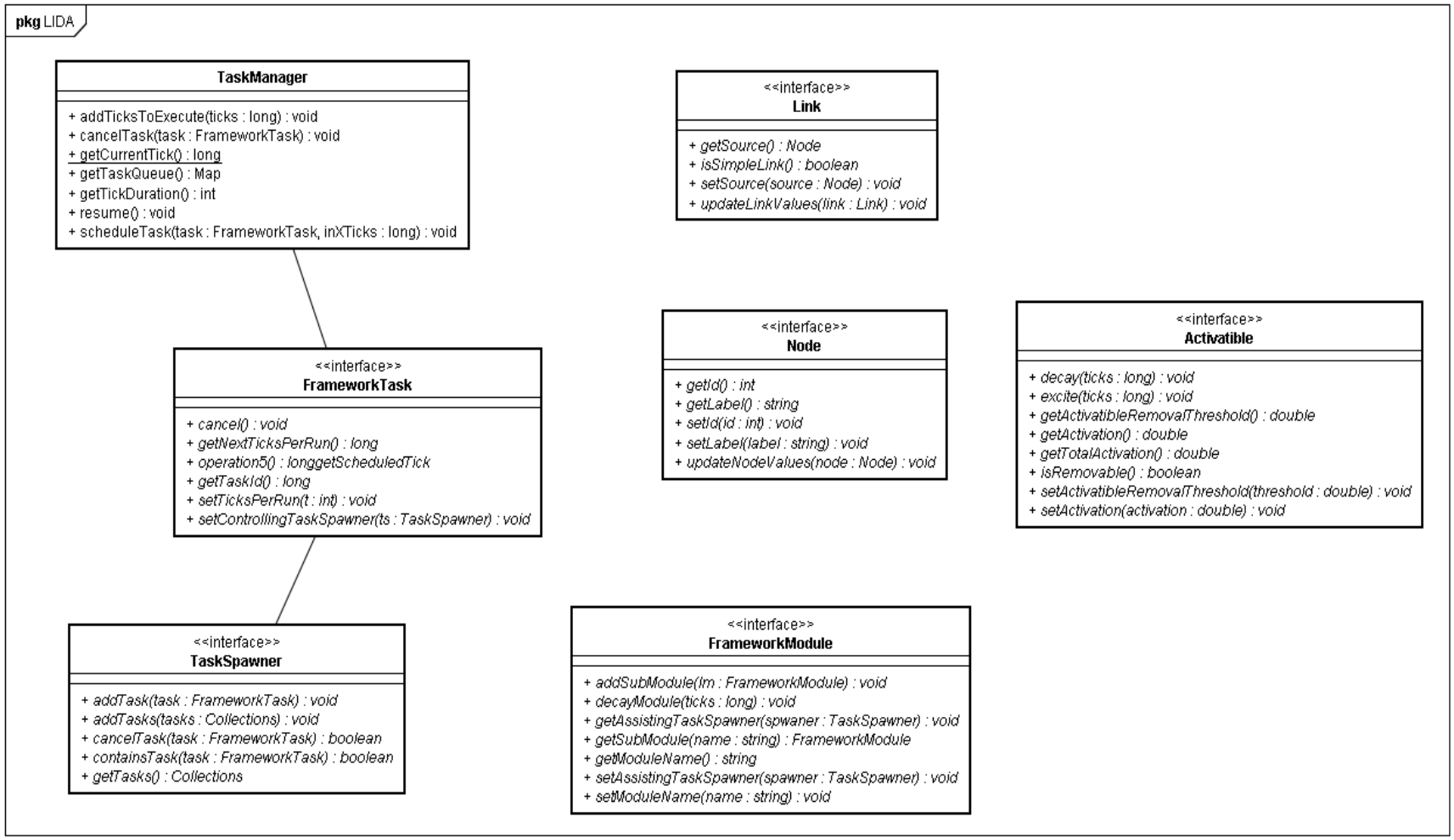
LIDA Framework

- Conceitos Gerais
  - Os diferentes Modules, utilizam-se em seu funcionamento, de diferentes redes, que possuem um formato normatizado
  - Para implementar essas redes, existem as classes **Node**, **Link** e **NodeStructure**
  - Os diferentes nós da rede possuem um nível de ativação
  - Para modificar esse nível de ativação, esses nós implementam a interface **Activatible**, que apresenta basicamente os métodos **excite** e **decay**
  - O framework permite a criação de estratégias de ativação, que são definidas em classes especializadas, chamadas **strategies**
  - Os nós que são passíveis de aprendizagem, devem implementar a interface **Learnable**, e incluem a definição de um base-level activation

LIDA Framework

- Conceitos Gerais
  - A configuração dos módulos a serem utilizados em uma aplicação do framework é realizada por uma série de arquivos de configuração:
    - Arquivo de configuração do framework: lidaConfig.properties
    - Arquivo de configuração do agente: arquivo XML contendo os modules a serem utilizados e as classes que os implementam
    - Arquivo de configuração das redes: arquivo XML contendo as definições de nós, links, estratégias e tarefas
    - Arquivos de configuração do uso da GUI: arquivos do tipo property, com diversos parâmetros de configuração do uso da GUI
  - A inicialização do agente é realizada por meio da classe **AgentStarter**, que se encarrega de carregar todos os arquivos de configuração e inicializar o agente

LIDA Framework - Overview



GROUP FOR RESEARCH ON ARTIFICIAL COGNITION

LIDA Framework

- Arquivo de configuração: lidaConfig.properties

```
#Agent properties
lida.agentdata=configs/alifeAgent.xml
lida.elementfactory.data=configs/factoryData.xml

#Gui properties
lida.gui.panels=configs/guiPanels.properties
lida.gui.commands=configs/guiCommands.properties
lida.gui.enable=true

#Logging properties
lida.logging.configuration=configs/logging.properties
```

UNICAMP

LIDA Framework: Configuração do Agente

```
<lida xmlns="http://ccrg.cs.memphis.edu/LidaXMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://ccrg.cs.memphis.edu/LidaXMLSchema LidaXMLSchema.xsd ">
    <taskmanager>
        <param name="taskManager.tickDuration" type="int">1 </param>
        <param name="taskManager.maxNumberOfThreads" type="int"> 100</param>
    </taskmanager>
    <taskspawners>
        <taskspawner name="defaultTS">
            <class>edu.memphis.ccrg.lida.framework.tasks.TaskSpawnerImpl</class>
        </taskspawner>
    </taskspawners>
    <submodules>
        <module name="Environment">
            <class>myagent.modules.ButtonEnvironment</class>
            <param name="height" type="int"> 10 </param>
            <param name="width" type="int">10 </param>
            <taskspawner>defaultTS</taskspawner>
        </module>
        <module name="SensoryMemory">
            <class>myagent.modules.ButtonSensoryMemory</class>
            <associatedmodule>Environment</associatedmodule>
            <taskspawner>defaultTS</taskspawner>
            <initialTasks>
                <task name="backgroundTask">
                    <tasktype>SensoryMemoryBackgroundTask</tasktype>
                    <ticksperrun>5</ticksperrun>
                </task>
            </initialTasks>
        </module>
        <module name="PerceptualAssociativeMemory">
            <class>edu.memphis.ccrg.lida.pam.PerceptualAssociativeMemoryImpl</class>
            <param name="pam.Upscale" type="double"> 7 </param>
```

LIDA Framework: Configuração das Redes

- Factory Data
  - Strategies
    - Estratégias de ativação gerais utilizadas nos diferentes Modules, envolvendo decaimento, excitação, etc
  - Nodes
    - Diferentes tipos de nós, utilizados nos diferentes Modules
  - Links
    - Diferentes tipos de links, utilizados nos diferentes Modules
  - Tasks
    - Diferentes tarefas (codelets) utilizados nos diferentes Modules

LIDA Framework

- A detecção de certos tipos de comportamentos, o ambiente de simulação e a percepção dos dados é feita “via código”. Exemplo:

```
public class ColorFeatureDetector extends BasicDetectionAlgorithm{

public void init() {
    super.init();
}

public double detect() {
  int color = (Integer)
  sensoryMemory.getSensoryContent("visual",smParams);
  //cosine is a better comparison
  if(soughtColor == color){
  return 1.0;
  }
 return 0.0;
 }
```

LIDA Framework

- Modules
  - Environment
  - SensoryMemory
  - PerceptualAssociativeMemory
  - TransientEpisodicMemory
  - DeclarativeMemory
  - Workspace
  - PerceptualBuffer
  - EpisodicBuffer
  - BroadcastQueue
  - CurrentSituationalModel
  - AttentionModule
  - StructureBuildingCodeletModule
  - GlobalWorkspace
  - ProceduralMemory
  - ActionSelection
  - SensoryMotorMemory
  - Agent
  - UnnamedModule

- P/ cada Module
  - class
  - associatedmodule
  - Parâmetros
  - TaskSpawner
  - initialTasks
  - InitializerClass
- Listeners
  - Descrevem a comunicação entre Módulos

LIDA Framework

- O Module *Environment*
  - Representa o ambiente onde o framework sensoreia e atua
  - Pode ser simulado, ou funcionar como um “front-end” para o ambiente real, como por exemplo o hardware de um robô
  - Dois métodos principais:
    - **getState**: obtém o estado do ambiente, aceitando possivelmente parãmetros para a leitura de variáveis específicas
    - **processAction**: atua sobre o ambiente
- O Module *SensoryMemory*
  - Usualmente também é implementado pelo usuário do framework
- Outros Modules
  - Possuem implementação default, mas podem ser modificados pelo usuário do framework

Maiores Informações

- Referência LIDA:
  http://ccrg.cs.memphis.edu/tutorial/introduction.html

- The LIDA Framework as a General Tool for AGI – Parte 1
  http://www.youtube.com/watch?v=Rgjw8O3vLBs