Aula 12 – LIDA 1

Atividade 1 – Estudo do Ciclo Cognitivo

A proposta deste tutorial fornecido pelos desenvolvedores da arquitetura LIDA é auxiliar um aprendizado menos profundo do que o tutorial completo poderia proporcionar, de acordo com os interesses do estudante.

O autor inicia afirmando que todo agente autônomo seja ser vivo ou artificial necessariamente opera em um ciclo de percepção, cognição e ação. Ele vai ainda mais longe, defendendo a hipótese de que um ser humano executa operações utilizando um ciclo cognitivo parecido com o da arquitetura em uma frequência de cinco a dez ciclos de operações assíncronas por segundo. A seguir, serão destacados conceitos relativos a este ciclo cognitivo, extraídos da exploração do fluxograma interativo fornecido no tutorial.

1) Percepção

É um processo pré-consciente onde ocorre a percepção de elementos do ambiente como categorização e identificação de objetos, relações situações, sentimentos e eventos. O processo ocorre em dois níveis.

O baixo nível é processado na chamada Memória Associativa Perceptual (*Perceptual Associative Memory*) ou *Slip Net*. Informações internas do agente ou aquelas colhidas pelos sensores externos são armazenadas na Memória Sensorial (*Sensory Memory*). Lá, são identificadas características básicas como cores, texturas, formas e movimento. Esta memória alimenta a PAM no caminho para processamento cognitivo (‘consciente’) e a Memória Sensorial-Motora (Sensory-Motor Memory) que opera sem utilizar mecanismos de consciência. A Slip Net é uma rede semântica com ativação onde nós são os elementos percebidos (características, objetos, categorias, situações, etc). Ela atua como um filtro, guardando apenas informações consideradas relevantes que constituem a **percepção atual** do agente, que é utilizada pelo Workspace.

O alto nível da percepção é responsabilidade do chamado *Workspace,* que consiste nos buffers pré-conscientes da memória de trabalho da arquitetura e inclui a muitas vezes chamada Memória de Longa Duração. No Workspace se encontram as percepções atuais do agente, percepções anteriores que ainda não expiraram, estruturas construídas pelo próprio Workspace para compreensão de conceitos de alto nível, associações recuperadas da Memória Episódica Transiente (*Transientic Episodic Memory*) ou da Memória Declarativa (*Declarative Memory*) que sejam atuais ou que ainda não tenham expirado.

2) Recuperação da Memória

Os elementos existentes no Workspace são utilizados para que buscas sejam feitas na memória episódica[1] do sistema. LIDA possui dois tipos de memória episódica: a Memória Episódica Transiente (Transient Episodic Memory) e a Memória Declarativa (Declarative Memory). A diferença básica entre elas é a duração de seus elementos. A memória transiente é alimentada pelo Workspace Global e seus elementos tem curta duração. A memória declarativa é alimentada através da consolidação dos elementos da MET e seus elementos persistem possível mas não necessariamente por toda a vida do agente, podendo expirar depois de um período de tempo assim como os elementos da MET, mas com uma persistência maior.

3) Difusão Consciente (*Concious Broadcast*)

É uma hipótese da teoria do Workspace Global na qual *codelets de atenção*[2] colhem informações do Workspace e competem para leva-las à consciência. Esta 'competição' ocorre no Workspace Global e a informação do 'vencedor' seria então transmitida largamente pelo aparato cognitivo de forma a possibilitar a seleção de recursos apropriados para lidar com a situação atual. Desta forma, o Workspace Global age, assim como na percepção, como um filtro onde apenas o que for de maior importância para o agente é levado à 'consciência'.
Apesar de diversos recursos poderem ser selecionados para lidar com esta situação, o alvo principal da transmissão é a memória procedural, onde as informações podem ser utilizadas para selecionar a próxima ação a ser executada. Considera-se esse mecanismo como o início de consciência de fenômenos.

4) Ação

O mecanismo de ação age sobre o conteúdo atual da consciência (que ainda não decaiu) e levando em consideração os objetivos atuais do agente (sensações enviadas à consciência através de difusão consciente) e a percepção do ambiente (também levada consciência através da difusão consciente). A cada ciclo cognitivo, exatamente uma ação é selecionada.

Atividade 2 – Tutorial e Compreensão da Arquitetura LIDA

Introdução:

[1] Memória Episódica é um tipo de memória que armazena eventos (o que, quando e onde)
[2] Codelets de atenção trabalham separadamente tentando encontrar no workspace informações que possam ser utilizadas para realizar uma tarefa.

A primeira observação importante feita no tutorial é a distinção entre dois conceitos relacionados porém distintos: o modelo LIDA e o Framework de mesmo nome. O modelo é uma concepção teórica do funcionamento da mente e a relação entre seus diversos módulos, enquanto o Framework é uma das muitas possíveis implementações deste modelo na forma de um framework desenvolvido na linguagem JAVA.

Serão cobertos neste tutorial as técnicas básicas para design e implementação de agentes baseados no Modelo LIDA utilizando o Framework LIDA, começando com detalhes de alto nível do Modelo e passando para os detalhes dos módulos, processos e estruturas de dados do Framework.

O Modelo LIDA

Este modelo é baseado no teoria do Workspace Global de Baars e na existência do Ciclo Cognitivo, que defende que todo agente autônomo necessariamente atua em ciclos onde percebe o ambiente e selecionando uma ação a tomar. Agentes mais sofisticados como os seres humanos processam as informações sensoriais percebidas de forma a facilitar a seleção de sua próxima ação.

Depois que o ambiente é percebido e o sistema atualiza sua representação interna do mundo, um processo competitivo descrito pela Teoria do Workspace Global decide que aspecto desta representação requer atenção mais urgente. Com esta seleção o agente então seleciona uma ação apropriada a tomar e a executa, terminando o ciclo.

A Arquitetura LIDA

Devido à grande complexidade de se implementar agentes inteligentes, a utilização de um Framework que implementa e abstrai suas funcionalidades principais facilita a criação de agentes customizados sem a necessidade de implementar suas funcionalidades de baixo nível.

A maior parte dos módulos do modelo LIDA é implementada pela arquitetura mas classes abstratas são fornecidas para criação das partes genéricas da arquitetura de um agente. Os módulos são encapsulados e altamente desacoplados. Desta forma, para implementar um agente, é possível utilizar módulos como fornecidos pela arquitetura ou substituir alguns deles ou todos por aqueles que implementarmos.

Na arquitetura são identificados os seguintes elementos:

a) Módulo: Coleção de dados e processor para operar sobre eles
b) Tarefa: Algoritmo curto que pode ser executado repetidamente para implementar um pequeno processo
c) Gerenciador de Tarefas: Suporta execução multithreaded de diversas tasks
d) GUI configurável: exibe o estado interno atual dos processos e dados do sistema

e) Fábrica: utilizada para criar estruturas de dados e estratégias comuns
f) Estratégias: Encapsula algoritmos comuns
g) Interpretador XML: utilizado para carregar e criar um agente através de um arquivo XML

Conceitos Básicos da Arquitetura LIDA

a) Módulos

São componentes do modelo conceitual LIDA. Exemplos são a Memória Sensorial, Memória Associada Perceptual e o Workspace. Enquanto alguns módulos são genéricos outros são dependentes do agente ou problema em questão. O Workspace, por exemplo, pode ser o mesmo para inúmeros tipos diferentes de agentes, enquanto a memória sensorial e a memória senso-motor são bem específicas uma vez que variam, por exemplo, com o tipo de agente e a forma que percebe a atua no ambiente.
A arquitetura permite que criemos implementações das interfaces dos módulos fornecidos ou que estendamos a implementação padrão. Neste último caso, pode ser necessário que diversas funcionalidades sejam implementadas na classe herdeira para que o módulo funcione com o comportamento desejado.

b) Listeners

Para a finalidade de transmissão de informações entre módulos, o modelo utiliza o padrão *Observer Pattern*, que se baseia na transmissão de eventos através de *listeners*. Desta forma, o módulo sujeito dispara eventos que os módulos que se registraram como ouvintes têm a capacidade de receber. Para poderem se registrar como ouvintes, estes módulos necessitam implementar todos os métodos abstratos da interface Listener.
Este padrão de implementação permite que módulos sujeitos consigam transmitir eventos de forma genérica para qualquer módulo registrado como ouvinte. Contudo, módulos que possuem maior acoplamento podem utilizar o conceito de associação. Nesta relação, um módulo associado a outro pode utilizar *codelets* para trocar informações com módulo a que se associou mais diretamente.

c) Ciclo de execução

A organização e agendamento de tarefas dos diversos módulos da arquitetura são responsabilidade de um componente *singleton* chamado *TaskManager*. Basicamente, neste componente são enfileiradas tarefas geradas por *TaskSpawners* de cada módulo. Uma tarefa por sua vez é algo similar aos *Demons* da teoria do pandemônio de Jackson ou aos *codelets* de Hofstadeter. Elas podem são agendadas de forma que podem ser executadas uma ou diversas vezes de acordo com a necessidade. Cada tarefa é executada em sua própria *thread* no *tick* para o qual foi agendado. Um *tick* é uma

unidade de tempo que representa um ciclo de execução do *TaskManager* e a cada *tick* mais de uma tarefa pode ser executada assincronamente. Durante sua execução, a tarefa tem seu status atualizado e quando for finalizada ela é reenviada juntamente com seu resultado para o *TaskSpawner* que a criou para que quaisquer ações necessárias possam ser tomadas.

d) *Nodes* e *Links*

Os conceitos de nós e conexões da arquitetura LIDA são partes fundamentais da Memória Associativa Perceptual (PAM ou *Sliptnet*). A Slipnet é composta por um conjunto de nós (nodes) relacionados através de conexões (links) formando uma *NodeStructure*. Uma *NodeStructure* possui os meios para inserção, recuperação, remoção e gerenciamento dos nós e conexões.

Tanto nós quanto conexões possuem um nível de ativação que pode ser excitado a qualquer momento e que decai ao longo do tempo. Como dito anteriormente, um nó pode representar atributos, conceitos, sentimentos, e objetos, por exemplo, enquanto links representam o tipo de relacionamento existente entre os elementos que conecta (LinkCategory). Se um link conecta um nó a outro ele é chamado simples e se conecta um nó a um link é chamado complexo.

Assim como muitos outros componentes da arquitetura, nós e links podem ser estendidos para fim de implementação específica de funcionalidades requeridas pela aplicação.

É importante levar em consideração no momento do desenvolvimento do sistema que quando um nó é adicionado a uma *NodeStructure* uma cópia do mesmo é criada na mesma e então retornada para prevenir que o mesmo nó coexista em mais de um módulo. A cópia possui a mesma identificação do nó que a originou mas suas propriedades e funcionalidades podem ser alteradas sem que o nó original o seja.

e) Níveis de ativação

Nós e as conexões são exemplos de elementos que possuem um nível de ativação: números reais com valores variando de 0 a 1, aumentando quando o elemento é excitado e diminuindo quando este decai. Estes níveis podem considerados, de forma superficial, como a relevância de um elemento em um determinado estado do sistema através tempo e quando a ativação se torna menor que um determinado limiar o elemento removido. A forma que esta variação ocorre é determinada pelas chamadas estratégias (Strategy) que regulam quanto este valor diminui quando decai e quanto aumenta quando é excitado e são características de elementos da interface *Activable*. O framework provê estratégias default de excitação e decaimento caso o desenvolvedor do sistema deseje utilizar.

Existe uma extensão da interface *Activable* chamada *Learnable*. Elementos que implementam esta interface possuem um nível de ativação base que é principalmente

utilizado para a implementação de aprendizado (representando por exemplo a utilidade de um nó em alguma situação anterior que se assemelhe ao estado atual). Esta interface é muito importante uma vez que a arquitetura ainda não possui algoritmos de aprendizagem.

f) Criação de elementos

Para a criação de elementos como nós, conexões, tarefas, estratégias e outros, recomenda-se utilizar o *singleton ElementFactory*. Objetos de diferentes classes podem ser criados através deste mesmo componente, cada qual com suas configurações particulares.

Utilizando o *ElementFactory*, muitos valores default podem ser fornecidos de forma a se obter facilmente muitas das funcionalidades que os elementos de um sistema necessitam sem que seja necessário compreender em detalhes o processo de criação.

Inicialização do Framework

A arquitetura LIDA permite que configurações do ambiente de execução, do agente e dos elementos que o compõe sejam definidos em arquivos de configuração. Existem três arquivos de configuração que podem ser utilizados:

1. Arquivo de configuração primária

   Possui o caminho dos outros arquivos de configuração e propriedades como a que define se a interface gráfica deve ou não ser inicializada

2. *Factory Data File*

   Referenciado pelo arquivo de configuração primário, é um arquivo XML que define quais variações de elementos (nós, conexões, estratégias e tarefas) devem ser carregados e disponibilizados na execução da simulação além dos que são disponibilizados por default.

3. *Agent Declaration File*

   Referenciado pelo arquivo de configuração primário, é um arquivo XML que define:
   1) Parâmetros globais do processo de inicialização,
   2) A onfiguração do *TaskManager*
   3) Os tipos de *TaskSpawners* disponíveis
   4) As especificações das classes e parâmetros necessários para a criação do agente, incluindo configurações de associação entre módulos, as tarefas iniciais do agente, e as classes responsáveis por inicializar cada módulo
   5) A definição e configuração das conexões entre os módulos

4. *GUI Configuration Files*
   São dois arquivos referenciados pelo arquivo de configuração primário que contém as configurações para a interface da GUI e os comandos que neles podem ser executados. Este arquivo só é levado em consideração se a propriedade lida.gui.enable do arquivo de configuração primário possuir valor *true.*

Quando o agente é inicializado, ele recorre ao arquivo de configuração primário para localizar os arquivos secundários. De posse destes, utiliza o *Factory Data File* para carregar os tipos de elementos disponíveis durante a execução, cria um novo agente utilizando as definições existentes no *Agent Declaration File* (através da classe *AgentXMLFactory)*, cria a interface gráfica utilizando o arquivo de configurações da mesma (caso esteja habilitada no parâmetro citado do arquivo de configuração primário), carrega o agente e finalmente carrega e exibe a GUI.

Segundo o ideal de generalização dos componentes de LIDA, é fornecida a interface *Initializable*, que permite que o desenvolvedor implemente ou sobrescreva o método *init* chamado no momento de inicialização do elemento. Dentro deste método, o uso do método *getParam* permite acesso aos parâmetros dos arquivos de configuração. Muitos elementos default fornecidos pelo Framework implementam esta interface para que sejam inicializados.

Lidando com diferentes ambientes e agentes

Cada simulação requer seu próprio tipo de ambiente e agente. Para que isto seja possível, o Framework provê uma interface, *EnvironmentImpl*, que serve como interface entre o agente e o ambiente. No mínimo, a implementação desta interface deve incluir os métodos *getState* e *processAction*, que traz informações sensoriais e envia comandos de ação sobre o ambiente.

Uma consequência dos diferentes meios e conteúdos de percepção é a necessidade da implementação de uma memória sensorial específica para o domínio da aplicação. Existe um plano de uma implementação de uma memória sensorial padrão na arquitetura, mas enquanto isso não é feito, deve-se estender a classe *SensoryMemoryImpl* para lidar com as particularidades de cada ambiente.

Atividade 3 – Pesquisa dos Módulos de LIDA

1) Arquitetura Baseada em Codelets

No contexto do modelo LIDA, um codelet é um processo especializado e ativo cuja função deve ser definida em poucas linhas de código. Estes *codelets* agem de forma semelhante aos chamados *demons* da teoria do Pandemonium de Oliver Selfridge: agem independentemente verificando se determinadas condições são atingidas para realizar alguma ação. No contexto LIDA, geralmente estas condições são existência de uma determinada configuração de nós e links no grafo do *Perceptual Buffer* do *Workspace* e a ação normalmente consiste em gerar uma coalisão destes nós (diferentes tipos de *codelets* podem formar coalisões seguindo regras diferentes) ao *Workspace Global*, onde competirão pela difusão consciente. Exemplos de tipos de *codelets* são: Codelets de ateção, de comportamento, de expectativa, de informação e de intenção.

2) Behavior Network

É uma ferramenta, fundamentalmente baseada no modelo de rede de comportamento de Maes, utilizada pelo modelo IDA para selecionar um comportamento entre aqueles ativados pela memória procedural devido uma difusão consciente. Comportamentos são tipicamente ações simples que o sistema motor pode realizar. São compostos por um procedimento a executar assim como suas precondições, pós condições e ativação. Os nós da rede são comportamentos e links entre os nós são criados levando-se em conta estas pré-condições e pós-condições. Se a pós-condição de um nó corresponde à pré-condição de outro então existe um link sucessor entre eles. Existem também links indicadores de conflito, que liga dois nós onde a pós-condição do primeiro causa a negação da pré-condição do segundo. A cada ciclo cognitivo, a ativação de um nó é passada a seus sucessores e antecessores até que se atinja estabilidade.

3) Global Workspace Theory

É uma teoria proposta por Bernard Baars que tenta explicar ou modelar o comportamento da mente. De maneira superficial, ela se baseia na hipótese de que diferentes processos independentes e paralelos competem entre si no aparato cognitivo, de forma que a cada ciclo um se destacaria e tornaria seu conteúdo aquele da consciência.

Baars utilizou a metáfora do teatro para explicar o funcionamento de sua teoria. Nesta, existem um palco, uma plateia, pessoas (que podem estar na plateia ou atuando no palco) e um foco de luz. O conteúdo da consciência são os atores que estão sob o foco de luz que é direcionado pela atenção. Tanto os atores fora do foco de luz quanto a plateia assistem aqueles sob o foco. Se estimulada pelos atores sob o foco de luz, uma pessoa na plateia poderia subir ao palco e realizar sua performance. Atores no palco competem pelo foco de luz.

No contexto de LIDA, as pessoas são codelets, o palco é o Workspace e o foco de luz o Workspace Global.