Desenvolvimento de um protótipo de braço robótico integrado a um agente BDI

Pedro K. B. Kano[1], **Maiquel de Brito**[2]

[1]Universidade Federal de Santa Catarina (UFSC)
Joinville – SC – Brasil

[2]Universidade Federal de Santa Catarina (UFSC)
Blumenau – SC - Brasil

pedro.doka2012@gmail.com, maiquel.b@ufsc.br

Abstract. *Certain applications, such as smart factories in Industry 4.0, require devices that autonomously interact with the physical world and are capable of reacting and adapting appropriately to changes in the system while maintaining their long-term objectives. BDI agents are a suitable metaphor for developing systems with these characteristics. This article describes the implementation of a robotic arm, typical of industrial applications, that is controlled by a BDI agent. The features of autonomy, proactivity, and reactivity are experimentally evaluated based on the system's behavior*

Resumo. *Certas aplicações, como por exemplo as fábricas inteligentes da Indústria 4.0, requerem dispositivos que atuem sobre o mundo físico de forma autônoma e que sejam capazes de reagir e adaptar-se adequadamente a mudanças no sistema mantendo seus objetivos de longo prazo. Agentes BDI são uma metáfora adequada para o desenvolvimento de sistemas com estas características. Este artigo descreve a implementação de um braço robótico, típico de aplicações industriais, que é controlado por agente BDI. As características de autonomia, proatividade e reatividade são avaliadas experimentalmente a partir do comportamento do sistema.*

1. Introdução

Sistemas robóticos (ou simplesmente *robôs*) são essenciais nos processos de manufatura das fábricas inteligentes (ou *smart factories*) da Indústria 4.0 [Rojko 2017]. Nelas, é esperado que os robôs colaborem uns com os outros para realizar tarefas que seriam difíceis ou até mesmo impossíveis para um único robô. Essa atuação deve ser realizada sem a intervenção de um operador externo e deve ser reconfigurável, de forma que os diferentes robôs adaptem-se a mudanças nas circunstâncias vigentes na linha de produção [Henning Kagermann 2013]. Para prover essas características às fábricas inteligentes, os robôs precisam ser (i) *autônomos*, para atuarem sem a intervenção de um operador externo; (ii) *socialmente capacitados*, para colaborarem entre si nos diferentes processos de manufatura; (iii) *proativos*, para desempenharem tarefas orientados a objetivos de longo prazo, que se mantém mesmo que as condições de operação se modifiquem; e, ao mesmo tempo (iv) *reativos*, para adaptarem-se tempestivamente a novas condições de operação [Aliyuda 2016].

Agentes – e, em particular, aqueles que seguem a arquiteura BDI – são uma metáfora computacional adequada para o desenvolvimento de sistemas com as características mencionadas anteriormente [Rao and Georgeff 1995, Wooldridge 2009]. No entanto, a utilização desta metáfora no desenvolvimento de robôs requer que os mecanismos de percepção e ação destes agentes sejam integrados elementos que compõem o *hardware* do robô. Este artigo explora o uso de ferramentas de programação orientada a agentes e de integração entre agentes e *hardware* para a construção de robôs que apresentem autonomia, habilidades sociais, proatividade e reatividade. Para isso, descreve-se a construção de um braço robótico cujo comportamento é especificado na forma de um agente BDI. Este braço robótico faz parte de um sistema mais amplo, em que interage com outros agentes e no qual precisa atuar de forma autônoma, proativa e reativa. As características de autonomia, proatividade e reatividade são avaliadas experimentalmente a partir do comportamento do sistema.

2. Fundamentação

Um *agente* pode como definido por sistema situado em um ambiente do qual tem percepções e no qual atua de forma autônoma para satisfazer os objetivos para os quais foi projetado [Wooldridge 2009]. Múltiplos agentes podem compor um Sistema Multiagente (SMA). Neste tipo de sistema, os agentes podem interagir uns com os outros para satisfazer seus objetivos. A *arquitetura* de um agente define os componentes que o compõem e a forma como estes componentes interagem entre si para que o agente raciocine, decida e atue [Wooldridge 2009]. Neste trabalho, considera-se especificamente a arquitetura BDI, segundo a qual um agente atua a partir dos seguintes elementos [Bratman et al. 1988, Rao and Georgeff 1995]:

- Crenças (Beliefs): informações que o agente possui sobre o mundo em que se encontra;
- Desejos (Desires): estados do mundo que o agente gostaria de (mas não está necessariamente atuando para) atingir;
- Intenções (Intentions): estados do mundo que o agente está atuando para atingir.

AgentSpeak é uma linguagem baseada em lógica para o desenvolvimento de agentes usando construtores de alto nível baseados na arquitetura BDI [Rao 1996], tais como crenças, objetivos e planos. *Jason* é um interpretador para *AgentSpeak* que inclui funcionalidades para comunicação entre agentes, permitindo desse modo a interação entre agentes e, por consequência, a implementação de SMA [Bordini et al. 2007]. Com a utilização do *Jason*, é possível criar agentes programando-se crenças, objetivos (equivalentes a desejos do modelo BDI) e planos de ação. A partir destes elementos, o interpretador da linguagem continuamente atualiza crenças dos agentes, avalia seus objetivos, produz intenções a partir dos objetivos factíveis, seleciona planos para satisfazer as intenções e os coloca em execução. Dessa forma, é possível desenvolver agentes que tenham (i) autonomia, para agir independentemente, sem a necessidade da intervenção ou supervisão de humanos ou outros sistemas; (ii) proatividade, para tomar a iniciativa sobre as ações a serem realizadas para atingir seus objetivos; (iii) reatividade, para adaptar-se a mudanças no sistema; e (iv) habilidades sociais, para se comunicar e cooperar com outros agentes.

O *framework embedded-mas*[1] fornece extensões para que agentes Jason sejam integrados a dispositivos físicos dotados de sensores e atuadores. Com essas extensões, as

[1] Disponível em `https://github.com/embedded-mas/embedded-mas`

percepções adquiridas pelos agentes incluem valores obtidos através de sensores físicos e o repertório de ações do agente inclui aquelas habilitadas por atuadores físicos. As percepções são obtidas a partir dos sensores físicos de maneira *passiva*, de modo que o agente não precisa atuar para obter valores dos sensores. Ao contrário, modificações nestes produzem percepções de forma automática. As ações dos agentes que são habilitadas por atuadores físicos são ações internas (ou *internal actions*). Assim, fazem parte da implementação do próprio agente em vez de serem tratadas como ações executadas sobre um dispositivo externo a ele.

3. Desenvolvimento

Para a análise das características buscadas (autonomia, proatividade, reatividade e habilidade social), foi montado um cenário em que um braço robótico está disposto entre um *buffer* e um *rack*, interagindo com um veículo autônomo (ou *delivery robot*), como ilustrado na Figura 1. O braço robótico é projetado como um agente BDI. Ele tem o objetivo de manter o *buffer* vazio, movendo os objetos postos sobre ele até o *rack*. Caso o rack já contenha um objeto, o braço robótico deverá primeiro enviá-lo através do *delivery robot*, para então esvaziar o buffer. Caso o *delivery robot* esteja carregando um objeto sobre si, ele pode solicitar que o braço robótico o remova. O braço robótico então passa a ter o objetivo de retirar o objeto do *delivery robot* e levá-lo para o *rack*, caso este não esteja ocupado. Caso o *rack* esteja ocupado, o braço robótico envia uma mensagem ao *delivery robot*, informando que que não será possível atender a sua solicitação.

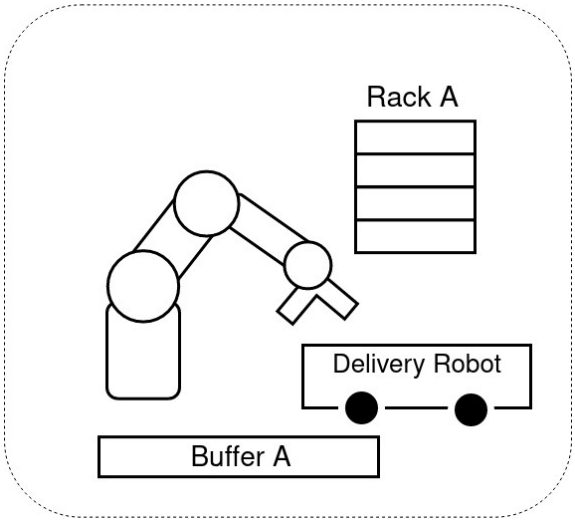

Figura 1. Representação do cenário em que o agente atuará

O braço robótico é construído a partir de um *kit* de robótica. Sua estrutura é composta de peças de acrílico cortadas a laser, contando com quatro micro servo motores 9g SG90, um microcontrolador Raspberry Pi Pico e uma fonte de alimentação externa. Os sensores de detecção de objetos no *buffer* e no *rack* são simulados através de botões. O braço robótico pode se movimentar entre os mais variados pontos, sendo restringido apenas pela limitação angular de cada micro servo motor. Para o cenário analisado, definiu-se quatro pontos em que o braço robótico será capaz de se posicionar: posição inicial, posição de retirada de itens do *buffer*, posição de depósito e retirada de itens do *rack* e posição de depósito e retirada de itens sobre o *delivery robot*. Além da movimentação

realizada por três dos servos motores que define a posição do braço robótico, haverá a movimentação da garra realizada pelo servo motor restante. Diferente do braço robótico, o *delivery robot* é simulado por um agente sem um correspondente físico real.

A estrutura física do braço robótico é integrada a um agente desenvolvido em *Jason*. Um microcontrolador *Raspberry Pi Pico*, que controla os sensores e atuadores do dispositivo, é conectado à porta serial de um equipamento com mais recursos, em que o agente *Jason* é executado. Este equipamento é, tipicamente, um computador pessoal ou um *single board computer*, como, por exemplo, *Raspberry Pi*. A cada ciclo de raciocínio do agente, os valores adquiridos pelos sensores são lidos pelo seu sistema de percepções e transformados em crenças. As ações do agente que são efetivamente realizadas através de atuações do braço robótico são transmitidas ao microcontrolador, que manipula os atuadores adequadamente.

O comportamento do agente é implementado usando elementos disponibilizados pelo *Jason*, que são, basicamente, crenças, objetivos, planos e ações internas. A Figura 2 exibe as principais partes desta implementação. Em sua atuação, o agente considera as seguintes crenças, obtidas através de percepções ligadas aos sensores físicos:

- *full(buffer)* e *empty(buffer)*, representando o *buffer* ocupado e vazio, respectivamente;
- *full(rack)* e *empty(rack)*, representando o *rack* ocupado e vazio, respectivamente;

O objetivo de manter o buffer vazio é representado, no código do agente, pelo objetivo *emptyBuffer*. O código do agente inclui planos para satisfazer este objetivo em diferentes circunstâncias. Entre elas, estão (i) o *buffer* vazio, quando agente espera o *buffer* estar cheio para seguir buscando atingir o objetivo *emptyBuffer* (Figura 2 - linhas 3 a 5), e (ii) o *buffer* cheio, em que o agente passa a ter o (sub)objetivo *move_buffer*, e, depois de satisfazê-lo, permanece buscando atingir o objetivo *emptyBuffer* (Figura 2 - linhas 7 a 9). Para satisfazer o objetivo *move_buffer*, o agente (i) move o item do *buffer* para o *rack* caso este último esteja vazio (Figura 2 – linhas 11 a 12) ou (ii) retira o objeto do rack, envia uma mensagem ao *delivery robot* requerendo o transporte do objeto e, finalmente, move o objeto do buffer para o rack (Figura 2 – linhas 13 a 16). O agente mantém o objetivo *emptyByffer* mesmo após executar os planos para satisfazê-lo (linhas 5 e 9). Por fim, o braço robótico pode receber requisições do *delivery robot* para retirada de um objeto, passando a ter o objetivo *remove_delivery*. A partir de tal objetivo, o braço robótico (i) envia uma mensagem ao *delivery robot*, solicitando que este se posicione para a retirada do item, e retira o objeto do *delivery robot* (Figura 2 – linhas 17 a 19) ou (ii) avisa o *delivery robot* que o *rack* está cheio e não será possível atingir tal objetivo no momento (Figura 2 – linhas 20 a 21) . O repertório de ações do agente inclui as ações *capt_B*, *capt_C* e *capt_R*, que consistem, respectivamente, em mover o braço robótico do local do *buffer* até o local do *rack*, mover o braço da posição em que o *delivery robot* se encontra até o *rack* e mover o braço robótico da posição do *rack* até o local do *delivery robot*. Estas ações são efetivamente realizadas através de atuações habilitadas pelos atuadores físicos do equipamento e, assim, são incorpradas à ação interna *defaultEmbeddedInternalAction* (Figura 2 – linhas 12, 14, 16, 19). Esta ação interna recebe três parâmetros: (i) o identificador do microcontrolador que controla o atuador que realizará a ação; (ii) o idenficador da ação a ser realizada, e (iii) uma lista de parâmetros necessários à realização da ação. Por exemplo, conforme a linha 12, ação *capt_B*, que não tem parâmetro algum, será realizada por meio de um atuador controlado por um microcontrolador identificado como *device1*.

```
1  !emptyBuffer. //initial goal
2
3  +!emptyBuffer : empty(buffer)
4    <- .wait(full(buffer));
5       !emptyBuffer.
6
7  +!emptyBuffer : full(buffer)
8    <- !move_buffer;
9       !emptyBuffer.
10
11 +!move_buffer : empty(rack)
12   <- .defaultEmbeddedInternalAction("device1","capt_B",[]).
13 +!move_buffer : full(rack)
14   <-  .send(delivery,achieve,move_delivery);
15       .defaultEmbeddedInternalAction("device1","capt_B",[]).
16
17 +!remove_delivery: empty(rack)
18   <- .send(delivery,achieve,move_delivery)
19      .defaultEmbeddedInternalAction("device1","capt_C",[]).
20 +!remove_delivery: full(rack)
21   <- .send(delivery,tell,full(rack)).
```

Figura 2. Trecho do código do agente

O código ilustrado na Figura 2 é integrado à porção física do agente, composta pelo braço robótico, através de uma especificação complementar feita em formato YAML [Telang 2020]. A Figura 3(a) ilustra partes desta especificação. Ela inclui definições sobre a conexão serial entre os sistemas de percepção e ação do agente e o microcontrolador (linhas 1 a 4) e a relação entre as ações que compõem o repertório do agente e as atuações implementadas pelo *hardware* (linhas 5 a 11). Por exemplo, a ação *capt_B*, que compõe o repertório do agente, é realizada pela atuação *captBuffer*, implementada no microcontrolador. A Figura 3 (b) ilustra parte da implementação desta atuação no código que é executado no microcontrolador. Se o agente executa a ação *capt_B*, então o microcontrolador recebe, via porta serial, o sinal de que a atuação *captBuffer* deve ser executada (Figura 3 (a) – linhas 6, 7 e Figura 3 (b) – linha 3) e, nesse caso, manipula os atuadores adequadamente Figura 3 (b) – linhas 3 a 12). [2]

4. Conclusão

A partir dos experimentos, observa-se que o braço robótico, cujo comportamento é implementado na forma de um agente BDI, comportou-se de forma autônoma, sem requerer a intervenção de um operador externo. Além disso, foi possível especificar um comportamento proativo para o braço robótico, em que ele toma a iniciativa de atingir um objetivo e atua constantemente para tal, mesmo que as circunstâncias do sistema se modifiquem (Figura 2 – linhas 3 a 9). Este comportamento proativo combina-se com um comportamento reativo, pois, ao buscar atingir tal objetivo, o agente adapta-se a mudanças no sistema. Por exemplo, se o *buffer* modificar seu estado de *vazio* para *cheio*, o agente modifica o plano que está sendo executado para atingir o objetivo *emptyBuffer* (Figura 2 – linhas 3, 4 e 7). Por fim, o braço robótico tem habilidades sociais, pois interage com o *delivery robot* para satisfazer seus próprios objetivos e também para satisfazer objetivos delegados por aquele agente. Assim, conclui-se que a programação orientada a agentes BDI, e, em particular, o uso do *Jason*, aliado a ferramentas de integração entre os sistemas de percepção e atuação

[2] O código do experimento está disponível em https://github.com/embedded-mas/raspberryPico-devs/tree/main/arm_buffer_rack

```yaml
microcontroller:
    id: device1
    serial: "COM3"
    baudRate: 9600
serialActions:
    - actionName: capt_B
      actuationName: captBuffer
    - actionName: capt_R
      actuationName: captRack
    - actionName: capt_C
      actuationName: captCar
```

(a)

```
while(Serial.available() > 0){
   String s = Serial.readString();
   if(s.equals("captBuffer")){
    for(pos = 110; pos >= 5; pos -= 1){
      rpServo1.write(pos);
      delay(15);
    }
    for(pos = 30; pos <= 120; pos += 1){
      rpServo2.write(pos);
      delay(15);
    }
    . . .
  }
}
```

(b)

Figura 3. Trecho de configuração da porção física do agente (a). As ações do agente são realizadas através de atuações implementadas no microcontrolador (b).

do agente e os sensores e atuadores do *hardware*, permitem o desenvolvimento de robôs com características essenciais às fábricas inteligentes da Indústria 4.0, que são autonomia, capacidade de interagir com outros robôs, proatividade e reatividade.

Referências

[Aliyuda 2016] Aliyuda, A. (2016). Towards the design of cyber-physical system via multi-agent system technology. *Int. Journal of Scientific & Engineering Research*, 7(10).

[Bordini et al. 2007] Bordini, R. H., Hübner, J. F., and Wooldridge, M. J. (2007). *Programming Multi-Agent Systems in AgentSpeak Using Jason*. J. Wiley.

[Bratman et al. 1988] Bratman, M. E., Israel, D. J., and Pollack, M. E. (1988). Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4:349–355.

[Henning Kagermann 2013] Henning Kagermann, Wolfgang Wahlster, J. H. (2013). Recommendations for implementing the strategic initiative industrie 4.0. Technical report, National Academy of Science and Engineering – Germany.

[Rao 1996] Rao, A. S. (1996). Agentspeak(l): Bdi agents speak out in a logical computable language. In Van de Velde, W. and Perram, J. W., editors, *Agents Breaking Away*, pages 42–55, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Rao and Georgeff 1995] Rao, A. S. and Georgeff, M. P. (1995). BDI agents: From theory to practice. In Lesser, V. R. and Gasser, L., editors, *Proceedings of the First International Conference on Multiagent Systems, June 12-14, 1995, San Francisco, California, USA*, pages 312–319. The MIT Press.

[Rojko 2017] Rojko, A. (2017). Industry 4.0 concept: Background and overview. *International Journal of Interactive Mobile Technologies (iJIM)*, 11(5):pp. 77–90.

[Telang 2020] Telang, T. (2020). *Introduction to YAML: Demystifying YAML Data Serialization Format*. Amazon Digital Services LLC - Kdp.

[Wooldridge 2009] Wooldridge, M. J. (2009). *An Introduction to MultiAgent Systems, Second Edition*. Wiley.