Concepção e análise de um modelo de agente BDI voltado para o planejamento de rota em um VANT

Fernando Rodrigues Santos[1], **Jomi Fred Hübner**[1], **Leandro Buss Becker**[1]

[1]Departamento de Automação e Sistemas
Universidade Federal de Santa Catarina (UFSC)
Caixa Postal 476 – 88040-900 – Florianópolis – SC – Brazil

fernando.rod.santos@gmail.com, {jomi.hubner, leandro.becker}@ufsc.br

Resumo. *O uso de agentes em aplicações com Veículos Aéreos Não-Tripulados (VANTs) tem sido explorado nos últimos anos, principalmente como alternativa para dotar o veículo de autonomia em suas missões. Este trabalho tem como objetivo desenvolver um modelo de comportamento autônomo para um VANT com o uso de um agente com arquitetura BDI, explorando sua capacidade de de reagir rapidamente a mudanças em seu ambiente e de ter objetivos de longo prazo. O artigo também discute a implementação do agente no sistema embarcado de um VANT real, além de apresentar uma comparação deste sistema com uma abordagem tradicional de programação.*

1. Introdução

Existem inúmeras possibilidades de aplicações usando Veículo Aéreo Não-Tripulado (VANT), como o monitoramento de regiões por times de VANTs, filmagens de shows, entre outras, como abordado em [Fahlstrom and Gleason 2012]. Em algumas situações, é necessário ou desejável, que o veículo tenha autonomia para realizar algumas tarefas. No contexto deste trabalho, um veículo é considerado autônomo quando ele não precisa da operação constante de um humano e ao qual possa ser atribuído objetivos. O processo de desenvolvimento de um VANT apresenta diversos desafios, como apresentados em [Gonçalves 2014], e a programação da autonomia deve levar em consideração as restrições de hardware, como desempenho, custo e consumo de energia do sistema embarcado do VANT, as quais limitam o uso de técnicas complexas que exijam grande capacidade de processamento da plataforma.

Na área de Sistemas Multi-Agentes (SMA), o tema autonomia é estudado há vários anos, produzindo teorias, arquiteturas de software e linguagens de programação voltadas especificamente para o desenvolvimento deste tipo de software, chamado de agente autônomo, segundo [Wooldridge 2002]. Como vantagens deste tipo de linguagem, vale destacar o seu alto nível de abstração, através de primitivas como objetivos, planos e ações que permitem ao desenvolvedor definir claramente o comportamento e a autonomia do agente. Nesta área, se destacam as linguagens baseadas na arquitetura BDI (Belief, Desire, Intention) que facilitam o desenvolvimento de agentes que apresentam a capacidade de reagir rapidamente a mudanças em seu ambiente e de possuir objetivos de longo prazo, conforme [Rao and Georgeff 1995] e [Bratman 1987].

A robótica móvel de modo geral, não só aplicações com VANTs, trabalha com planejamento de rotas, conforme [Romero et al. 2014], onde o planejador de rotas é o módulo responsável por determinar de forma autônoma o caminho a ser seguido pelo

robô, de acordo com a tarefa atualmente em execução. Dessa forma, busca-se uma solução com o desenvolvimento de um sistema de planejamento de rota para robôs móveis com o uso de agentes, visando explorar as potencialidades da arquitetura BDI nessas aplicações.

Assim, a proposta deste trabalho consiste em analisar a possibilidade do uso de um agente com arquitetura BDI no contexto de um sistema computacional de tempo real embarcado em um robô móvel, um VANT, para aplicações de tomadas de decisões e planejamento de rota. Além disso, é realizada uma comparação dessa abordagem proposta com uma abordagem tradicional de programação orientada a objetos.

No restante deste artigo expomos os trabalhos relacionados, apresentamos o modelo de comportamento autônomo para tomada de decisão de um robô móvel desenvolvido com um agente BDI, bem como uma visão geral do planejamento de rota em robótica móvel, além de um cenário de teste e a modelagem do problema. Também mostramos a sua implementação no sistema embarcado de um VANT e as análises desse sistema em comparação com uma abordagem tradicional de programação, por meio de testes de simulação.

2. Trabalhos Relacionados

O uso de SMA em veículos autônomos não é uma novidade, nota-se que o paradigma de agentes tem sido empregado em projetos com VANTs e que a maioria das abordagens apresentam soluções com agente e SMA. Alguns trabalhos modelam um VANT como um SMA, outros trazem resultados de coordenação em ambiente simulado, enquanto outro realiza testes com um VANT real acoplado a um simulador. Esses trabalhos exploram as potencialidades do SMA para tomada de decisão em grupo de VANTs, e em algumas aplicações utilizam outras técnicas computacionais ou específicas da área do problema.

O trabalho de [Hama 2012] aborda uma aplicação do paradigma da programação orientada a agentes para controle inteligente do comportamento de VANTs, com a concepção de um framework, através do uso da arquitetura, da teoria e de ferramentas orientadas a agentes, que visa prover uma abstração para a programação de comportamentos inteligentes em VANTs. Naquele trabalho é proposto o modelo UAVAS - Unmanned Aerial Vehicles AgentsSpeak, que é um framework de programação de comportamentos para VANTs, que é executado dentro do sistema operacional do hardware embarcado da aeronave. Aquele trabalho utiliza um sistema com múltiplos agentes (SMA) para controlar um único VANT, sendo que ainda enfatiza a comunicação entre os agentes e a cooperação do time. Tem-se o uso de agentes BDI, mas o foco é dado no modelo de abstração. Porém, o framework não foi implementado e testado em uma aplicação embarcada real, somente em simulação.

No trabalho de [Chaves 2013], foram estudados diferentes algoritmos de navegação e padrões de busca adequados, bem como uma visão geral sobre mecanismos de coordenação multi-agente para coordenação distribuída de VANTs, visando cooperação. O projeto propõe um modelo de VANTs cooperativos que combina mecanismos de coordenação multi-agente, algoritmos de navegação para varredura completa de uma região de busca e padrões estabelecidos pelos principais órgãos responsáveis pelas operações de busca e salvamento. Este trabalho explora a cooperação em grupo de VANTs com foco na coordenação do SMA e seus resultados são apenas em nível de simulações em PC, sem implementação em plataformas de VANTs reais, a qual dependerá da capaci-

dade do hardware utilizado para implementar e executar o modelo proposto.

Já o trabalho de [Selecký and Meiser 2012] aborda o uso de equipe com múltiplos VANTs autônomos. Este trabalho utiliza um simulador de VANTs e tráfego aéreo que permite trabalhar tanto com os VANTs virtuais quanto reais em um sistema de realidade mista, no qual VANTs reais e virtuais podem coexistir. O mesmo apresenta algumas soluções para os problemas no processo de integração de VANTs reais em um sistema de simulação multi-agente, que consiste em um ambiente de simulação onde os VANTs podem coexistir, coordenar o seu voo e cooperar em tarefas comuns. Esses VANTs reais são capazes de realizar planejamento on-board e raciocínio, e podem cooperar e coordenar seus movimentos uns com os outros, e também com os VANTs virtuais. Ele explora a parte de integração do VANT de hardware no ambiente simulado, mas não garante que o sistema possa ser utilizado sem modificações em uma aeronave real. Além disso, não investiga ou trata da possibilidade do uso de um agente em um VANT, utilizando uma estrutura do tipo HIL (Hardware in the Loop).

Logo, esses trabalhos não exploram o uso de um único agente BDI em um VANT e as potencialidades da arquitetura, tais como a sua capacidade de reagir rapidamente a mudanças no ambiente e de ter objetivos de longo prazo, buscando assim, dotar um VANT de autonomia em suas aplicações. Da mesma forma, não analisam a possibilidade da aplicação real da abordagem com agente BDI no sistema computacional embarcado de um VANT, sendo este o foco do presente trabalho.

3. Modelo de Planejamento de Rota

Esta seção apresenta uma visão geral do planejamento de rota em robótica móvel, além do modelo proposto com agente BDI. Também mostra o cenário de teste e a modelagem do problema, bem como a implementação e aplicação prática do modelo.

3.1. Visão Geral do Planejamento de Rota

Geralmente, a estrutura de controle de navegação de um Robô Móvel Autônomo (RMA) é organizada em cascata [Romero et al. 2014], com quatro níveis de controle. No nível 4 encontra-se o planejamento dinâmico e a geração das rotas do robô. No nível 3 são executados tipicamente os algoritmos de controle que são responsáveis pela condução do veículo, através do seguimento ou rastreamento de trajetórias, baseados em modelos cinemáticos ou em equações de movimento translacional, os quais garantem o movimento desejado para o robô. No nível 2 é executado o controle da dinâmica do robô, com o objetivo é manter as velocidades longitudinal e lateral do veículo, ou a rotação do robô, e suas derivadas, estabilizadas em torno de um ponto de equilíbrio, que mantém a estabilidade do movimento do veículo. No nível 1 são implementados os sistemas de controle dos sensores e atuadores, garantindo assim as percepções e atuações do robô no ambiente. A figura 1 ilustra o exemplo da estrutura de controle em cascata de um RMA.

O planejamento de rota consiste em dadas as configurações inicial e final do robô, descobrir uma sequência de movimentos a ser executada para que ele saia da primeira e atinja a segunda. Assim, a solução proposta neste artigo trabalha com o enfoque no nível 4 da estrutura de controle de navegação, para o desenvolvimento de um modelo de comportamento autônomo com um agente BDI para planejamento de rota e tomada de decisão em robôs móveis.

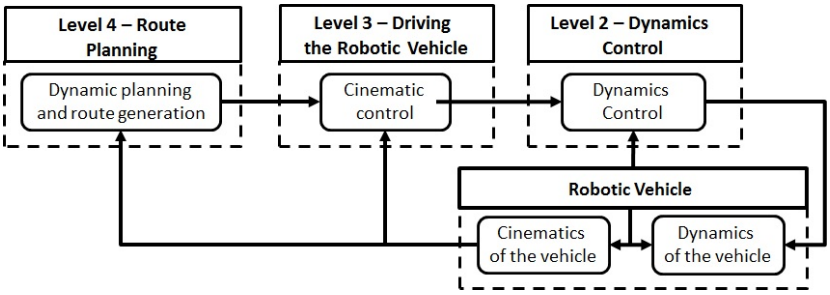

Figure 1. Estrutura de controle em cascata, adaptado de [Romero et al. 2014].

3.2. Modelo Proposto

O sistema proposto é um modelo de comportamento do VANT baseado no modelo de fluxo e conversão de dados para interação do agente com o ambiente proposto em [Hama 2012], porém com a utilização de um único agente BDI no VANT e não um SMA. A arquitetura do sistema é inspirada em [Chaves 2013], com dois níveis: um nível para os controles de estabilidade e de translação da aeronave; e outro nível para o planejamento de rota. Já a parte de testes, inspira-se em [Selecký and Meiser 2012] e seu modelo de integração, mas em vez da integração do VANT de hardware, fez-se uma integração da plataforma embarcada da aeronave com o modelo computacional do VANT.

A solução proposta com agente BDI trata do nível de **Planejamento** da estrutura do controle de navegação. Os controles da dinâmica e da condução do veículo foram desenvolvidos e implementados pelo projeto ProVant[1], sendo utilizados como parte do sistema proposto. Assim, os níveis 1, 2 e 3 foram encapsulados em um único nível, chamado de **Controle**, que executa as funcionalidades desses níveis, tendo uma nova estrutura como ilustrada na figura 2.

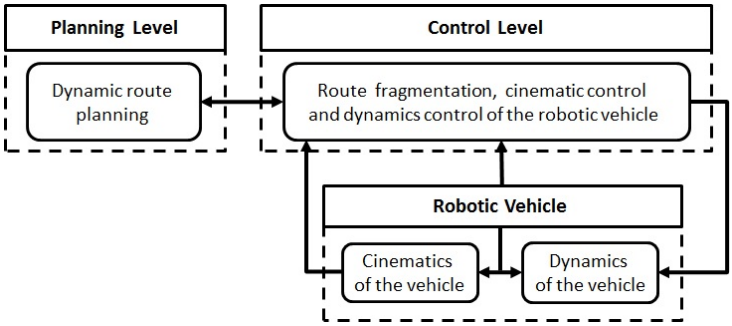

Figure 2. Estrutura proposta.

O fluxo de informação entre os níveis ocorre com os sinais dos sensores do veículo robótico indo do nível de **Controle** para o nível de **Planejamento**, fornecendo as informações das percepções do ambiente requeridas pelo agente, conforme a aplicação. Com base nesses dados, o nível de **Planejamento** realiza o seu processo de tomada de decisão, escolhendo um plano de ação com o ponto e a rota, além de definir as atuações a serem executadas no ambiente, enviando para o nível de **Controle** os pontos de destino (*waypoints*) da aeronave. Assim, o controle executa o algoritmo de translação e aciona os atuadores do sistema.

[1]Site: http://provant.das.ufsc.br

Uma interface de comunicação entre os dois níveis garante a troca de informações. O nível de **Controle** recebe as mensagens de atuação do nível de **Planejamento** com os valores de referência de velocidade e da posição de destino, e envia as percepções com as informações da posição atual (GPS), valores da carga e da taxa de consumo da bateria. A partir de uma nova mensagem do nível de **Planejamento**, ocorre a fragmentação da rota em pontos intermediários entre a posição atual do VANT e o *waypoint* de destino, através de um percurso retilíneo dividido em trechos menores com passo fixo. Também acontece a correção do ângulo de direção da aeronave em relação ao ponto a ser alcançado. Dessa forma, as referências de posição e ângulo são passadas para o nível de **Controle**, garantindo assim o seguimento da rota definida no nível de **Planejamento**.

No nível de **Controle** estão implementados os algoritmos de controle de estabilidade e de controle de translação do VANT, conforme detalhado em [Donadel et al. 2014]. Neste nível, são executadas as rotinas do controle translacional e do controle rotacional, com base nas velocidades em cada direção do eixo carteziano (x,y,z) e os respectivos valores de referências adotados. Além disso, também é realizada uma operação de transformação dos sinais de saída para as grandezas de medidas compatíveis com os atuadores do VANT, sendo estes, sinais de força e ângulo.

3.3. Cenário de Teste

O cenário escolhido para testar o sistema consiste em uma aplicação para visitar os nodos de uma rede de sensores sem fio com um VANT. Os sensores se encontram distribuídos numa região de monitoramento, a aeronave deve visitar todos os nodos da rede, a partir da estação-base, e retornar para a base finalizando a missão. A posição de cada sensor é conhecida pelo VANT desde o início da missão. A medida que a aeronave alcança um *waypoint*, ela tenta obter os dados daquele sensor e, em seguida, passa para o próximo nodo da rede.

A aeronave deve manter uma altitude de cruzeiro, levar em consideração a carga e a taxa de consumo de bateria do VANT, sendo que a taxa varia de acordo com a velocidade e direção do vento. A medida que a aeronave se desloca, mantendo uma velocidade constante de voo, a carga da bateria decrementa, conforme a direção do VANT e do vento. Se o vento está em sentido contrário à aeronave, a taxa de consumo de bateria é maior e com vento a favor o consumo é menor.

O VANT deve analisar também o "ponto de não retorno"[2], levando em consideração a carga e taxa de consumo da bateira, bem como a distância até a base. Quando o VANT perceber que não conseguirá visitar todos os nodos da rede devido a carga de bateria, ele deve abortar a missão de coleta de dados e retornar à estação-base para recarrega. Além disso, durante o retorno à base, caso o consumo de bateria seja alto e não seja possível chegar com segurança, o VANT deve realizar um pouso de emergência.

3.4. Modelagem

Com base no cenário de aplicação, foi realizada a modelagem do problema para ambas as abordagens, visando a comparação e análise do sistema.

[2]O ponto de não retorno é definido como aquele trecho da rota de um voo em que o combustível restante na aeronave não é suficiente para que ela retorne ao aeroporto de partida, em caso de emergência.

Para a abordagem com agentes, foi utilizado o método Prometheus AEOlus de [Uez 2013] para desenvolvimento da aplicação com um agente BDI, o qual tem como objetivo permitir o projeto e a análise de um sistema com agentes.

A modelagem do sistema foi feita com o Diagrama de Objetivos do Sistema, que permite a análise dos objetivos em forma de árvore de decomposição AND/OR, como ilustrado na figura 3. Esse diagrama descreve os objetivos do sistema durante a sua execução. Também foi feito o Diagrama de Visão Geral do Sistema, que permite a visualização da estrutura global do sistema e mostra além do agente, suas percepções e ações, bem como suas crenças iniciais, conforme a figura 4. Assim, tem-se o agente ***VANT*** com as percepções de posição, carga e taxa de consumo da bateria, bem como as atuções com a velocidade e a posição de destino da aeronave, além das crenças.

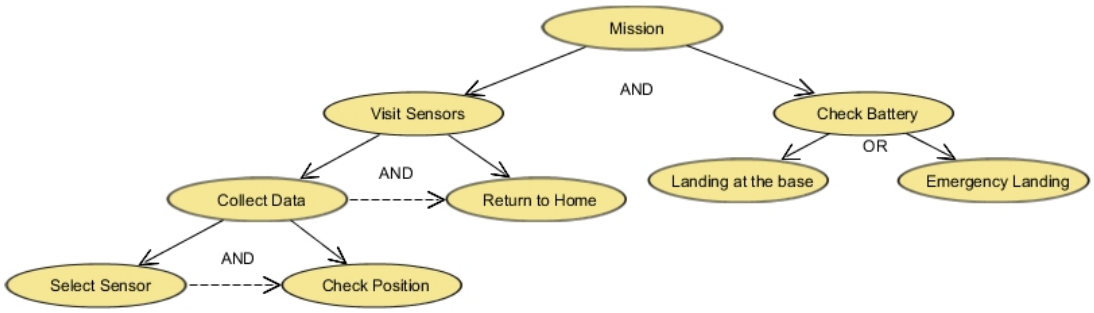

Figure 3. Diagrama de Objetivos do Sistema.

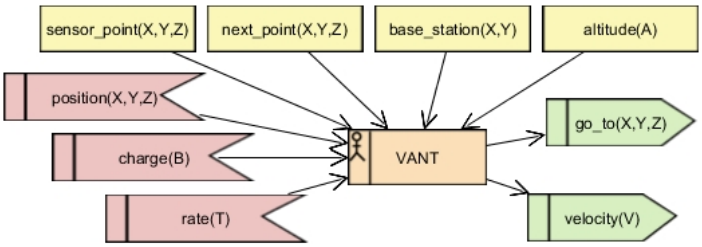

Figure 4. Diagrama Visão Geral do Sistema.

O agente possui como crenças a posição de todos os nodos sensores que precisa visitar (`sensor_point(X,Y,Z)`), a informação de qual será o próximo ponto a ser vistidado (`next_point(X,Y,Z)`), a localização da estação-base (`base_station(X,Y)`), além da altitude de cruzeiro (`altitude(A)`).

A modelagem do comportamento da aeronave para abordagem tradicional, foi realizada com a máquina de estados da figura 5, onde cada estado representa uma situação na qual se encontra o VANT em determinado momento da execução do sistema e as trasições correspondem aos eventos que acarretam em uma mudança de estado.

3.5. Implementação do Planejamento de Rota

O nível de **Planejamento** realiza o processo de planejamento de rota e de tomada de decisão do VANT. Neste módulo foi desenvolvido uma versão com o uso de um agente com arquitetura BDI e, para comparação, outra versão com programação orientada a objetos.

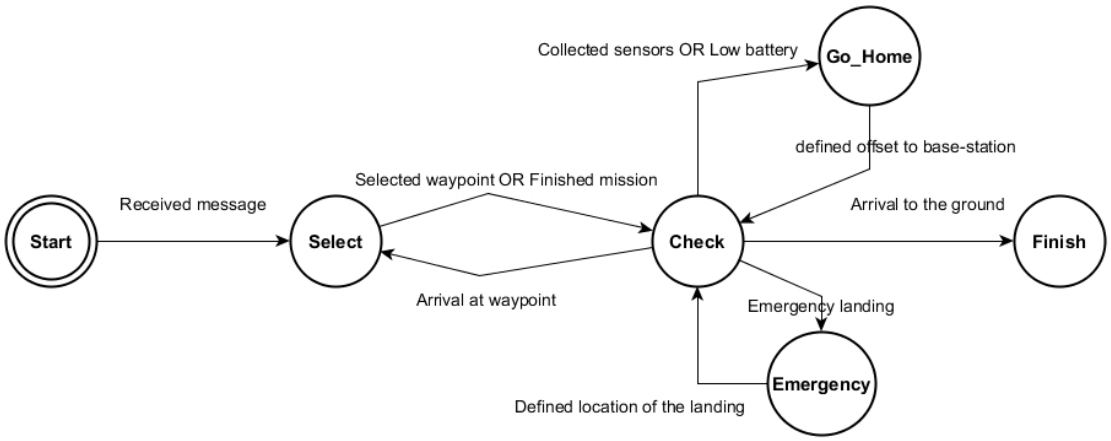

Figure 5. Máquina de Estados.

A versão com agente BDI foi implementada em linguagem Jason[3], tendo uma arquitetura customizada que permite implementar as percepções e as atuações no ambiente. Além disso, foi necessária uma infraestrutura de integração para que o código do agente possa acessar o código do sistema embarcado do robô móvel, viabilizando a integração entre o agente e seu ambiente. Enquanto o mecanismo recebe e mantém as informações mais recentes do ambiente, estas ficam disponíveis para o agente no momento em que ele decide realizar a percepção. Além disso, as informações de atuação do agente permanecem na interface até que uma nova decisão seja tomada para atuar no ambiente.

Um agente pode ser desenvolvido com dois estilos de programação, reativo e proativo. Nesta proposta, os planos em que o agente deve reagir rapidamente a mudanças no ambiente foram implementados de forma reativa, que ocorrem a cada nova percepção do ambiente ou alteração na sua base de crenças e intenções. Por exemplo, o plano de verificação da carga da bateria (programa 1) é executado quando o agente percebe a mudança no valor da carga da bateria (`+charge`), não possui energia suficiente para prosseguir a missão (`not energy`) e não possui o desejo de ir para base (`.desire(go_home)`). Assim, esse plano descarta todos os eventos e intenções do agente (`.drop_all_desires`) e cria a intenção (`!go_home`) de retorno para base.

Já os planos em que o agente tem objetivos de longo prazo e necessite de comprometimento com um plano até que ele seja finalizado foram implementados de forma proativa. Por exemplo, o plano de coletar dados dos sensores da rede (programa 2), baseado na especificação da figura 3 é executado quando o agente tiver essa intenção (`!collect_data`) e possuir sensores (`sensor_point(_,_,_)`) em sua base de crenças. Esse plano ativa outros planos através das intenções de seleção do próximo nodo sensor (`!select_sensor`) e da verificação da chegada ao local de destino (`!check_position`). Após atingir o *waypoint* alvo, o agente consulta o próximo ponto (`?next_point(X,Y,Z)`) que desejava alcançar e remove a informação do sensor (`-sensor_point(ID,X,Y)`) dessa posição. Então, retoma-se novamente a execução do plano com a ativação da intenção (`!collect_data`). Esse processo se repete até que todos os sensores da base de crenças do agente sejam visitados.

[3][Bordini et al. 2007] e Site: http://jason.sourceforge.net/wp/

Programa 1. Plano para verificação da carga da bateria (reativo).

```
+charge(_) : not energy & not .desire(go_home)
    <- .drop_all_desires;
       !!go_home.
```

Programa 2. Plano para coletar dados dos sensores da rede (proativo).

```
+!collect_data : sensor_point(_, _, _)
    <- !select_sensor;
       !check_position;
       ?next_point(X,Y,Z);
       -sensor_point(ID,X,Y);
       !collect_data.

+!collect_data.
```

Outra versão do nível de **Planejamento** foi desenvolvida utilizando uma abordagem tradicional, com programação orientada a objetos e implementada em linguagem C++. Esse sistema opera em um laço de repetição, com uma estrutura de seleção para execução de acordo com o estado do VANT. Neste laço, inicialmente se processa as mensagens com os valores de percepção, e então se realiza o processo de tomada de decisão e envio de mensagem com a atuação, segundo a especificação da figura 5. A tomada de decisão é feita por meio do cálculo das distâncias entre a aeronave e o ponto de destino.

4. Aplicação Prática

O modelo proposto foi implementado no sistema embarcado do VANT do projeto ProVant[4] para o cenário de aplicação descrito anteriormente. Esse veículo possui duas plataformas computacionais embarcadas que se comunicam pela porta serial, sendo uma de baixo nível (*discovery*) para acionamento dos sensores e atuadores, e outra de alto nível (*beaglebone*) para implementação do controle e planejamento da aeronave.

O sistema foi testado diretamente na plataforma de alto nível, a qual será embarcada na aeronave para execução dos níveis de **Planejamento** e **Controle**. Assim, para substituir o VANT real nos testes, foi utilizado um modelo computacional que possui o modelo matemático comportamental do VANT e representa a aeronave real da aplicação, fornecendo os dados dos sensores e sendo alimentado com as informações para os atuadores da aeronave, além das funções da plataforma de baixo nível. Este modelo computacional foi desenvolvido e implementado pelo projeto ProVant, em Matlab/Simulink, sendo este apenas utilizado como parte integrante do modelo final do sistema proposto.

Dessa forma, o sistema proposto foi implementado em um modelo de simulação Hardware In the Loop (HIL)[5] para a realização de testes e análises. A estrutura geral do modelo HIL é composta por uma plataforma embarcada com os níveis de **Planejamento** e de **Controle** que se comunicam via UDP, além de um computador com o modelo computacional da planta (VANT) e um mecanismo de comunicação entre as plataformas, por meio de um link TCP/IP. Tendo as respectivas estruturas para as abordagens com agente e tradicional, conforme ilustrado na figura 6.

[4]Site: http://provant.das.ufsc.br

[5]A simulação Hardware-in-the-loop (HIL) é descrita como a técnica aplicada ao desenvolvimento, teste e validação de sistemas de tempo real embarcados complexos. Este ambiente proporciona uma plataforma efetiva permitindo a adição de complexidade, com a segurança da plataforma de testes, conforme [Louall et al. 2011].

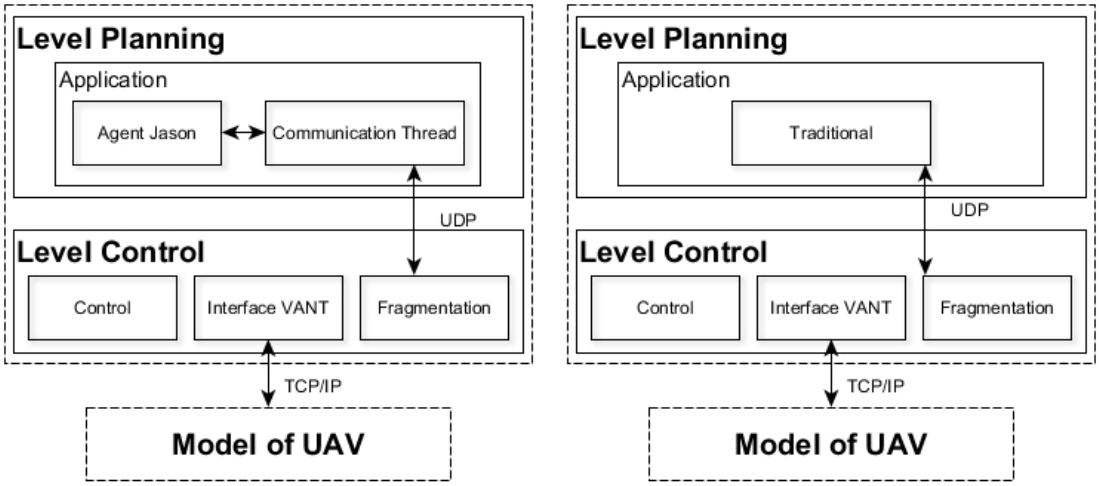

Figure 6. Estrutura HIL: abordagens com agente e tradicional.

Os ensaios com o modelo HIL realizados visaram testar o sistema proposto com agente BDI em uma plataforma embarcada. Dessa forma, não foram adicionados ruídos ao processo para testar estabilidade e robustez do sistema de controle.

5. Análises e Resultados

Como resultado deste trabalho, obteve-se duas versões do sistema de tomada de decisões e planejamento de rota de um robô autônomo. Esses sistemas foram implementados no sistema embarcado de um VANT e testados através de um modelo de simulação HIL que descreve o comportamento da aeronave.

A missão do VANT nas simulações foi de percorrer a área onde estão os nodos da rede de sensores sem fio, visitar cada um deles e retornar para estação-base, levando em consideração as variações do ambiente durante a sua execução.

5.1. Análise Qualitativa

A análise qualitativa busca fazer uma análise a nível de projeto e desenvolvimento do sistema com uso de uma linguagem de agentes em uma aplicação de sistemas embarcados, a qual geralmente se utiliza uma abordagem tradicional de programação e tem-se restrições de hardware e processamento.

5.1.1. Análises Iniciais

O desenvolvimento do agente BDI foi inicialmente baseado na implementação da abordagem tradicional, com a criação de flags de estado como crenças do agente. Porém, após a familiarização com a linguagem, foram realizadas melhorias e refinamentos no código, sendo este processo bastante custoso com relação ao tempo de projeto, até se chegar a uma implementação baseada em objetivos e não em estados.

Percebeu-se que o uso de uma linguagem de programação de agentes BDI facilita o desenvolvimento de agentes que necessitem de capacidade de reagir rapidamente a

mudanças no ambiente (como exemplifica o programa 1), bem como possuir objetivos de longo prazo em uma aplicação (como exemplifica o programa 2), conforme empregado na implementação dos planos do agente no sistema. Já uma abordagem tradicional apresenta dificuldade de se definir comportamentos proativos na modelagem em máquina de estados, além disso, o desenvolvimento desse tipo de comportamento requer a implementação de processos concorrentes, levando a uma programação mais complexa. Por outro lado, máquinas de estados são amplamente conhecidas e possuem várias ferramentas de análise, além de serem rapidamente e eficientemente implementadas.

Nota-se também que no uso de uma linguagem de programação de agentes, como Jason, muitas vezes, tem-se a necessidade do desenvolvimento de uma infraestrutura de integração para aplicações embarcadas. Visto que estas aplicações são desenvolvidas, geralmente, em linguagem C.

5.1.2. Diferença de programação

Analisamos a diferença de programação entre as abordagens para se inserir um novo comportamento ou funcionalidade no sistema. Na abordagem com agentes BDI, a inserção de um novo comportamento ou alteração de uma funcionalidade na aplicação é realizada através da criação de um novo plano, sem mexer no que já existe no código, como pode ser notado pela independência entre os códigos dos programas 1 e 2.

Já na abordagem tradicional de programação, desenvolvida com uma estrutura de seleção baseada nos estados do sistema, a inserção de um novo comportamento ou alteração de uma funcionalidade na aplicação é realizada através da criação de uma nova condição na execução do sistema como um todo. Além disso, deve-se verificar a lógica para que não haja conflito entre as demais condições para que não aconteçam simultaneamente. Logo, por serem paradigmas diferentes, apresentam características específicas que interferem no modo como programar o comportamento desejado para a aplicação.

5.1.3. Modelagem

Na análise da modelagem do problema, tem-se que a área de SMA não possui um método padrão para desenvolvimento de sistemas, nem um ferramental de desenvolvimento tão consolidado como para a abordagem tradicional. Além disso, cada abordagem modela o comportamento do sistema de uma forma particular, sendo que a Máquina de Estados especifica a sequência de estados pelos quais um processo passa durante o tempo de execução em resposta aos eventos, enquanto o Diagrama de Objetivos do Sistema descreve os objetivos do sistema durante a execução do processo.

Comportamentos sequenciais podem ser modelados tanto em uma como em outra abordagem, porém, os comportamentos proativos são mais difíceis de serem modelados em Máquinas de Estados. Enquanto comportamentos complexos e concorrentes podem ser modelados com o Diagrama de Objetivos do Prometheus AEOlus.

5.2. Análise Quantitativa

A análise quantitativa busca mensurar o desempenho das versões do sistema, visando comparar seus rendimentos em uma aplicação real. Duas formas de análise foram uti-

lizadas para avaliar os sistemas desenvolvidos: o tempo de utilização da CPU e o tamanho do código fonte gerado por cada abordagem. Para a análise do uso do processador, foram realizadas simulações do modelo HIL com as versões do sistema proposto.

5.2.1. Utilização da CPU

O tempo de utilização do processador pelo sistema de tomada de decisão no nível de **Planejamento** foi medido através do comando *time* do sistema operacional Linux. Esse comando informa o tempo de execução de um processo específico. Assim, dado o tempo total de execução da simulação, com o VANT percorrendo todos os sensores e retornando para estação-base, mediu-se qual a fatia de tempo que o processo de planejamento faz uso do processador para executar as suas tarefas. Com base no tempo total e no tempo do processo, obtem-se um percentual de uso da CPU pelo processo.

Os primeiros testes, foram realizados utilizando a comunicação com o envio da percepção para o nível de **Planejamento** a cada mudança na posição do VANT. Com isso, o agente sempre tinha uma nova informação para atualizar em sua base de crenças quando realizava o processo de percepção no seu ciclo de raciocínio. A abordagem com agente BDI teve um tempo de utilização da CPU de 61% e a abordagem tradicional teve um tempo 4%. Percebeu-se que o agente utilizava muito do seu tempo para fazer percepção, já que a frequência de atualização das informações é alta (a cada 5 ms).

Nos testes seguintes, foi utilizada a comunicação com o envio da percepção para o nível de **Planejamento** somente após uma mudança significativa na posição do VANT, sendo essa maior que 0,1m. Com isso, obteve-se uma redução do tempo de uso da CPU, tanto com o agente como o tradicional. Foram realizados experimentos com 10 amostras, os resultados do agente Jason obteve Média = 15,158 % e Desvio Padrão = 1,126. Já a abordagem tradicional obteve Média = 0,833 % e Desvio Padrão = 0,230. Assim, a solução com agente obteve um desempenho satisfatório, o que possibilita a sua execução numa aplicação em plataforma embarcada sem comprometer o desempenho do hardware.

5.2.2. Tamanho do Código

Outro ponto analisado foi o tamanho do código fonte, considerando somente a parte do código referente ao nível de planejamento do sistema, para cada uma das abordagens utilizadas.

Como a quantidade de linhas no código fonte de um programa depende do estilo de programação do desenvolvedor, optou-se por utilizar o comando *gzip* para compactar o arquivo fonte (sem comentários) dos programas e verificar o tamanho do mesmo em bytes. Além disso, também realizou-se a contagem do número de identificadores utilizados no código fonte de cada programa.

O código do agente Jason possui 0,700 Kbytes e 81 identificadores, enquanto o código da abordagem tradicional tem 2,164 Kbytes e 165 identificadores. Por ter um nível de abstração maior e utilizar a programação lógica, o código do agente é menor que o da versão tradicional.

6. Considerações Finais

Conclui-se que é possível a execução de um agente BDI embarcado em um VANT, tendo em vista a redução do percentual de uso da CPU apresentado nas análises e testes através de um modelo de simulação HIL. Embora necessite de uma infraestrutura para integração com o sistema embarcado do robô móvel, a solução com agente é viável para aplicações de planejamento de rota e pode ser embarcada em um robô autônomo.

O trabalho trouxe como contribuição um modelo de comportamento para planejamento de rota de um VANT autônomo com uma abordagem de agente BDI, além da integração do sistema na plataforma computacional embarcada da aeronave. Também apresentou algumas potencialidades da arquitetura BDI em VANTs, analisando algumas vantagens e desvantagens do uso desse tipo de agente nessas aplicações, podendo essa solução ser utilizada e explorada em outros cenários da robótica móvel, tais como veículos terrestres, aquáticos, entre outros.

Como trabalhos futuros, pretende-se extender a aplicação para o cenário mais completo de coleta de dados em uma rede de sensores sem fio, além da realização de testes do modelo proposto com o VANT real.

References

Bordini, R. H., Hübner, J. F., and Wooldridge, M. (2007). *Programming Multi-Agent Systems in AgentSpeak using Jason (Wiley Series in Agent Technology)*. Wiley-Interscience.

Bratman, M. E. (1987). *Intention, Plans, and Practical Reason*. Harvard University Press.

Chaves, q. N. (2013). Proposta de modelo de veículos aéreos não tripulados (vants) cooperativos aplicados a operações de busca. Master's thesis, USP - Universidade de São Paulo.

Donadel, R., Raffo, G. V., and Becker, L. B. (2014). Modeling and control of a tiltrotor uav for path tracking. *19th World Congress The International Federation of Automatic Control*.

Fahlstrom, P. G. and Gleason, T. J. (2012). *Introduction to UAV Systems*. Wiley.

Gonçalves, F. S. (2014). Projeto da arquitetura de software embarcado de um veículo aéreo não tripulado. Master's thesis, UFSC - Universidade Federal de Santa Catarina.

Hama, M. T. (2012). Uma plataforma orientada a agentes para o desenvolvimento de software em veículos aéreos não-tripulados. Master's thesis, UFRGS - Universidade Federal do Rio Grande do Sul.

Louall, R., Belloula, A., Djouadi, M., and Bouaziz, S. (2011). Real-time characterization of microsoft flight simulator 2004 for integration into hardware in the loop architecture. *Control Automation (MED) - 19th Mediterranean Conference*.

Rao, A. S. and Georgeff, M. P. (1995). Bdi agents: from theory to practice. *Proceedings of the First International Conference on MultiAgent Systems*.

Romero, R. A. F., Prestes, E., Osório, F., and Wolf, D. (2014). *Robótica Móvel*. LTC.

Selecký, M. and Meiser, T. (2012). Integration of autonomous uavs into multi-agent simulation. *Acta Polytechnica*, 52(5).

Uez, D. M. (2013). Método para o desenvolvimento de software orientado a agentes considerando o ambiente e a organização. Master's thesis, UFSC - Universidade Federal de Santa Catarina.

Wooldridge, M. (2002). *An Introduction to MultiAgent Systems*. Chichester: John Wiley and Sons Ltd.