André Dionísio Bettencourt da Silva Rocha

Licenciado em Ciências da Engenharia Electrotécnica e de Computadores

An Agent Based Architecture for Material Handling Systems

Dissertação para obtenção do Grau de Mestre em

Engenharia Electrotécnica e de Computadores

Orientador: José António Barata de Oliveira, Professor Doutor, FCT-UNL

Co-orientador: Luis Domingos Ferreira Ribeiro, Doutor, UNINOVA/CTS

Júri:

Presidente: Prof. Doutor Luis Filipe dos Santos Gomes

Arguente: Prof. Doutor Adolfo Steiger Garção

Vogais: Prof. Doutor José António Barata de Oliveira
Doutor Luis Domingos Ferreira Ribeiro

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro de 2013

An Agent Based Architecture for Material Handling Systems

Para Mafalda e Família

Agradecimentos

Nesta secção gostaria de expressar os meus agradecimentos a todos aqueles que contribuíram para a realização desta dissertação.

Quero agradecer em primeiro lugar ao meu orientador, Professor Doutor José Barata, por me ter dado a oportunidade de desenvolver esta dissertação no âmbito do projeto IDEAS. A possibilidade a mim endereçada, permitiu-me evoluir como pessoa e académico.

Ao meu co-orientador, Doutor Luís Ribeiro, um agradecimento muito especial. Por toda a disponibilidade, apoio e dedicação. Durante o ano e meio de trabalho, dirigido ao desenvolvimento deste projeto, muitas foram as horas passadas em conjunto de forma a alcançar o sucesso do mesmo.

Quero também agradecer ao Rogério Rosa, por toda a disponibilidade, aquando da minha chegada ao laboratório. Foi um dos pilares durante o desenvolvimento deste projeto.

Não posso deixar de demonstrar a minha gratidão para com Fábio Oliveira, sempre disponível em todos os momentos, sem exceções, e para com o José Gonçalves. Com eles passei muitos dos meus dias, durante o meu percurso académico.

Não podia deixar de referir também a minha gratidão para com a Lídia, por estar sempre disponível e por ter facilitado os meus primeiros tempos fora do ambiente onde nasci e cresci. Precioso apoio que nunca esquecerei.

Um agradecimento também especial ao Sr José Nascimento pelo apoio demonstrado e disponibilidade.

Não podia deixar de referir a minha gratidão para com José Manuel pela disponibilidade que se revelou preciosa no caminho percorrido.

Quero também agradecer à Anabela, Eduardo e Nuno por todo o apoio e por me terem recebido como o fizeram.

Um agradecimento especialíssimo para a Mafalda, por tudo aquilo que significa para mim e por toda a sua paciência durante este tempo de projeto, principalmente na fase final, em que a sua ajuda foi preciosa.

Por fim, agradeço à minha família, em especial aos meus pais, padrinhos, Gonçalo e tia Idalina, pela oportunidade concedida, por todo o apoio dado e pela confiança demonstrada. Permitindo que este percurso seja bem-sucedido.

A todos vós,

O meu sincero e caloroso Muito Obrigado!

Resumo

Num passado recente as exigências do mercado alteraram-se e como consequência as linhas de produção também. Com a customização de produtos e o crescente número de produtos a produzir, o dinamismo e a flexibilidade das linhas são agora exigências de extrema preponderância.

As abordagens tradicionais denotam grande dificuldade em satisfazer estas necessidades e como tal, têm aparecido algumas propostas de modo a suprimilas. As abordagens propostas são maioritariamente referentes ao escalonamento e planeamento da produção.

O sistema de transporte não é habitualmente inserido na arquitetura de controlo e reconfiguração do sistema, sendo constantemente posto de parte no que se refere a esta problemática.

Neste trabalho é proposta um arquitetura auto-organizada de suporte ao sistema de transporte, onde este desempenha funções de controlo e gestão do mesmo. A arquitetura foi desenvolvida para um sistema composto por passadeiras transportadoras onde as estações disponibilizam tarefas nas mesmas.

A arquitetura proposta é uma arquitetura multiagente onde é utilizado o algoritmo de *Dijkstra* para melhorar o encaminhamento de produtos e matéria-prima. A arquitetura tem como principaiscaraterísticas o balanceamento de carga presente nas passadeiras transportadoras como a capacidade de acoplar e desacoplar estações durante a execução do mesmo.

A arquitetura foi primeiramente testada num ambiente virtual de modo a serem verificados os comportamentos da mesma e posteriormente foi testada num célula industrial real, de modo a comprovar a sua utilização num sistema real.

Palavras-chave: Sistema de Transporte Multiagente, Balanceamento de Carga, Produção, Produto Ativo, Auto-Organização.

Abstract

In the recent past, market requirements and consequently the production lines changed too. With the customization of products and the growing number of products to produce, the dynamism and flexibility of the lines are now requirements of extreme importance.

A traditional approach indicates great difficulty in satisfying those needs and as such has appeared some proposals in order to solve them. The proposed approaches are mostly related to scheduling and production planning.

The transportation system is not usually inserted in the control architecture and system reconfiguration, constantly being put aside in order to this issue.

This work proposes architecture to support self-organized transportation system, where it performs control functions and management. The architecture was developed for a system comprising conveyors where the stations operate.

The proposed work is a multi-agent architecture that use Dijkstra's algorithm to improve the routing of products and materials. The main features of architecture are load balancing present in conveyors and ability to plug and unplug stations in runtime.

The architecture was first tested in a virtual environment in order to check the behavior of the same and was subsequently tested in a real industrial cell in order to demonstrate its use in a real system.

Keywords: Multi-Agent Transportation System, Load Balancing, Production, Active Product, Self-Organization.

Acrónimos

AD	Algoritmo de Dijkstra
BD	Base de Dados
BL	Balanceamento de Linhas
	(Assembly Line Balancing)
CP	Comprimento da Passadeira
FIPA	Foundation for Intelligent Physical Agents
HA	Habilidade Atómica
	(Atomic Skill)
HC	Habilidade Compexa
	(Complex Skill)
HUA	Handover Unit Agent
JADE	Java Agent Development Framework
NE	Número de Estações acopladas
NP	Número de Produtos no preciso momento
OCF	Ótimização por Colónia de Formigas
	(Ant Colony Optimization)
PA	Product Agent
PSBL	Problema Simplificado do Balanceamento de Linhas
	(Simple Assembly Line Balancing Problem)
PSBL-1	Problema Simplificado do Balanceamento de Linhas 1
	(Simple Assembly Line Balancing Problem 1)
PSBL-2	Problema Simplificado do Balanceamento de Linhas 2
	(Simple Assembly Line Balancing Problem 2)
PSBL-E	Problema Simplificado do Balanceamento de Linhas E
	(Simple Assembly Line Balancing Problem E)

PSBL-F	Problema Simplificado do Balanceamento de Linhas F
	(Simple Assembly Line Balancing Problem F)
PSiA	Product Sink Agent
PSoA	Product Source Agent
RFID	*Radio-Frequency IDentification*
	(Identificação por Rádio-Frequência)
SAT	Sistemas Automatizados de Transporte
	(Automated Material Handling Systems)
SBM	Sistemas Biónicos de Manufatura
	(Bionic Manufacturing Systems)
SEP	Sistema Evolutivo de Produção
	(Evolvable Production Systems)
SFM	Sistemas Flexíveis de Manufatura
	(Flexible Manufacturing Systems)
SHM	Sistemas Holónicos de Manufatura
	(Holonic Manufacturing Systems)
SMA	Sistema Multi Agente
	(Multi Agent System)
SRM	Sistema Reconfigurável de Manufatura
	(Reconfigurable Manufacturing Systems)
ST	Sistema de Transporte
TE	Tabela de Encaminhamento
TEA	Transport Entity Agent
VGA	Veículo Guiado Automático
	(Automated Guided Vehicle)
VP	Velocidade de transporte da Passadeira
YPA	Yellow Pages Agent

Índice de Conteúdos

Índice de Figuras

Índice de Tabelas

1.Introdução

1.1. Apresentação do Problema

Nos primórdios da produção, um pequeno número de modelos e possibilidades de construção faziam com que a produção em massa resultasse numa ótimização das linhas de produção, através da especialização da mão-de-obra.

Com o passar do tempo e com as exigências dos consumidores a alterarem-se e com as mudanças introduzidas pelos mercados o número de modelos e a quantidade de produtos a produzir de cada modelo alterou-se drasticamente. Estas alterações devem-se à crescente procura de modelos altamente customizados e personalizados, como também ao elevado número de diferentes produtos à disposição do cliente.

Sendo assim, a constante allteração de produtos a serem produzidos como as constantes variações das quantidades a serem produzidas, levam a que os sistemas de produção tenham novas exigências, como dinamismo e flexibilidade.

Com as abordagens tradicionais os sistemas tornaram-se aos poucos obsoletos e inadequados a esta nova realidade. Esforços têm sido realizados ao longo do tempo de modo a fazer frente a estas novas necessidades.

Várias abordagens são propostas, desde o reajuste dos sistemas mais conservadores até a soluções mais evolutivas. Estas soluções mais recentes surgiram com a aplicação da informática à indústria de várias maneiras.

O sistema de transporte presente na linha de produção habitualmente é excluído das tomadas de decisão no que se refere a balanceamento de carga e otimização da produção. De forma a integrar o sistema de transporte nas tomadas de decisão referentes à otimização, o trabalho realizado propõem uma arquitetura multiagente auto-organizada que prevê a incrementação do desempenho do sistema através das decisões tomadas por estas entidades próativas.

1.2. Pergunta de Investigação e Hipóteses

Tendo como base o problema anteriormente referido e sendo o sistema de transporte parte fundamental da linha de produção, algumas questões podem ser colocadas:

- Será possível tornar o sistema de transporte uma parte ativa na tomada de decisão do processo de produção no contexto dos sistemas de produção evolutivos e auto-organizados?

- Que tipo de arquitetura poderia suportar a formulação anterior do problema do transporte?

É proposta como hipótese o desenho de uma arquitetura multiagente auto-organizada de suporte ao sistema de transporte. A arquitetura proposta é capaz de tomar decisões baseadas nos estados dos seus elementos e dos restantes elementos do sistema, de forma a incrementar o desempenho do mesmo.

1.3. Visão Geral do Trabalho Realizado

Formuladas as questões anteriores é proposto o desenho de uma arquitetura auto-organizada baseada num sistema multiagente, de suporte a um sistema de transporte constituído por uma rede de passadeiras transportadoras.

A arquitetura proposta é constituída essencialmente por dois tipos de entidades, as que abstraiem os encaminhadores e as que abstraiem as passadeiras transportadoras. Cada uma das entidades tem um conhecimento global da rede, este conhecimento é adquirido gradualmente. A informação relevante à constituição da rede é partilhada entre os vários elementos do sistema de transporte, através do envio das tabelas onde indicam os vizinhos de cada um deles.

Sempre que uma nova entidade pretende associar-se ao sistema já em execução este deverá comunicar aos elementos vizinhos a sua vontade e as suas caraterísticas, de modo a que estes possam renovar a sua informação em relação a este novo elemento. Desta forma, a adição de novos elementos ao sistema durante a execução é verificada.

Cada passadeira transportadora tem caraterísticas inerentes a esta e estados durante a execução que são partilhados com os vizinhos. Com esta informação e recorrendo ao Algoritmo de *Dijkstra* é calculada em encaminhador a árvore de custos mínimos. Esta árvore de custos mínimos é frequentemente atualizada de maneira a que os estados da passadeira transportadora, como o número de estações presentes e o número de produtos a transportar, possam ser indicadores importantes no cálculo da mesma. Esta abordagem tem como objetivo o

balanceamento da carga através da associação de uma métrica que se rege pelos estados da passadeira transportadora.

A arquitetura proposta prevê também o acoplamento e desacoplamento de estações de trabalho durante a execução. Assim cada passadeira transportadora tem a habilidade de durante a execução receber uma nova estação. No acoplamento de uma nova estação a passadeira onde é associada tem a responsabilidade de comunicar a todos os restantes elementos do sistema a existência da nova estação naquele ponto do sistema de transporte. A partir desse momento todo o sistema é capaz de encaminhar produtos que requiram o encaminhamento até essa nova estação. O comportamento inverso também é verificado, onde o desacoplamenteo também é previsto e onde o sistema de transporte tem a capacidade de se auto-organizar de maneira a saber que aquela estação já não se encontra disponível.

Também é proposto na arquitetura a possibilidade de um operador puder remover ou adicionar um produto ao sistema durante a execução do mesmo. A remoção de produtos ou adição de mesmos é possível na passadeira transportadora, à exceção das estações. Quando um operador remove uma palete contendo um produto, este tem a capacidade de detectar essa saída, através da consulta do *Radio-Frequency Identification* (RFID) e reorganizar as suas filas de espera. O caso contrário também se verifica, quando se dá a entrada de um produto numa destas zonas da passadeira.

Posteriormente ao desenho da arquitetura esta foi implementada recorrendo à biblioteca JADE. Os testes realizados à arquitetura decorreram primeiramente em ambiente simulado, recorrendo-se a um simulador desenvolvido também em trabalho de Mestrado e posteriormente testada numa célula real.

1.4. Principais Contribuições

As principais contribuições do trabalho realizado prendem-se com o facto de se apresentar uma arquitetura que permite a otimização da produção recorrendo a elementos do sistema habitualmente descorados destas tarefas.

Sendo os elementos do sistema de transporte entidades extremamente ativas num ambiente de produção a sua utilização poderá trazer uma enorme mais-valia na resolução do problema de otimizaçãode sistemas altamente dinâmicos de uma forma auto-organizada e mantendo a natureza *plug-and-produce* dos mesmos.

A auto-organização que rege o sistema proposto possibilita que este seja proativa de forma a tomar as melhores decisões em cada momento no encaminhamento dos produtos na rede. Desta forma, em cada momento o sistema de transporte faz o encaminhamento que melhor faz o balanceamento da carga pelos vários elementos do transporte, fazendo com que a

alteração de produtos em processamento e alterações na topologia não se tornem fatores limitadores no sistema.

2. Estado da Arte

Originalmente as linhas de produção foram desenvolvidas tendo como objetivo uma produção em massa de produtos padronizados. Esta padronização visava a exploração da mão-de-obra altamente especializada e os conhecimentos associados a essa aprendizagem. No entanto, os requisitos dos mercados, refletindo-se também nos requisitos exigidos aos sistemas de produção, alteraram-se drasticamente. A customização extrema e a grande variedade de produtos exigem uma nova abordagem no contexto de produção.

O transporte de matérias-primas e produtos no ambiente de produção é normalmente suportado por uma rede de passadeiras transportadoras ou por um conjunto de veículos teleguiados automáticos (VGA). Em algumas situações verifica-se a existência de uma solução híbrida, com zonas onde o transporte é feito pelas passadeiras e noutras onde o transporte fica a cargo dos VGA's.

Estes elementos responsáveis pelo transporte de materiais raramente são incluídos no processo de decisão. O sistema de transporte habitualmente não é incluído na tarefa de equilibrar a produção entre os vários recursos disponíveis, sendo esta ação assegurada por um algoritmo de balanceamento de linhas que tipicamente ignora o custo de transporte.

2.1. Balanceamento de Linhas

O equilíbrio das linhas tem sido tradicionalmente abordado a partir de uma perspetiva de Balanceamento de Linhas (BL) ou *Assembly Line Balancing* na literatura. Segundo (Boysen, Fliedner et al. 2008) qualquer tipo de problema de BL consistem em encontrar um balanceamento de linhas possível, ou seja, cada uma das tarefas atribuídas a uma determinada estação tem de cumprir as limitações de precedência e todas as restrições adicionais.

Devido às repercussões a longo prazo das decisões de balanceamento, os objetivos desse balanceamento devem ser cuidadosamente escolhidos em função das metas estratégicas da empresa. Um objetivo frequentemente proposto é o de maximizar a utilização da linha, este aspeto é medido através da eficiência da linha.

2.1.1. O Problema Simplificado do Balanceamento de Linhas

O problema básico de BL é chamado de Problema Simplificado de Balanceamento de Linhas (PSBL) ou *Simple Assembly Line Balancing Problem* na literatura. De forma a resolver

problemas deste género foram inicialmente definidas quatro versões com diferentes objetivos de utilização. O PSBL-E tem como objetivo maximizar a eficiência da linha, frequentemente designada pela letra E. De forma a minimizar o número de estações, tendo como referência um tempo de execução tem-se o PSBL-1. O PSBL-2 tem o funcionamento contrário, ou seja, minimizar o tempo de execução sabendo o número de estações. Por último o PSBL-F tenta encontrar uma solução viável, dado o número de estações e o tempo de produção.

A utilização do PSBL só é viável fazendo algumas assunções limitadoras. Com essas assunções é possível reduzir a complexidade do problema da configuração da linha, passando-a para o núcleo do problema da atribuição das tarefas às estações. À medida que esta abordagem foi aplicada aos sistemas reais foi verificada que existia uma diferença enorme entre as caraterísticas descritas no paradigma e as caraterísticas dos sistemas reais. Estas diferenças deviam-se às enumeras combinações encontradas nos ambientes de produção, devido a aspetos técnicos e organizacionais, criando sistemas diversificados. De forma a aproximar esta abordagem aos sistemas reais foram adicionadas extensões à ideia original (Boysen, Fliedner et al. 2008).

Com estas extensões, o BL pode ser implementado de uma forma mais específica a cada um dos casos, de acordo com a constituição da linha e os objetivos da produção. Com as várias extensões o BL pode ser aplicado de acordo com as seguintes variantes.

2.1.2. BL Dependendo do Número de Modelos a Produzir

A primeira variante resultante das extensões introduzidas consiste num reajuste do BL de acordo com o número de modelos que são objetivo produzir. De acordo com (Boysen, Fliedner et al. 2008) existem três possibilidades para esta variante.

A alta customização de produtos torna as linhas de produção de um único produto cada vez menos atraentes. Embora a procura por esta configuração tenha diminuído, a sua utilização continua a ser requerida em casos específicos,onde as linhas de modelo único têm como objetivo o balanceamento de linhas deste género.

As linhas de modelos misturados surgem com o intuito de possibilitar a montagem de diferentes modelos de um produto na mesma linha. Apesar dos esforços realizados nesta variante de BL visando a versatilidade desta abordagem, tendencialmente o processo de produção necessita de grande homogeneidade. Com esta limitação o sistema não suporta variações da mistura de entrada de produtos na linha, possibilitando a montagem de vários modelos mas onde a mistura de produtos concluídos é constante.

Quando a homogeneidade de produtos à entrada da linha e os processos não são garantidos, a abordagem vista anteriormente não é aplicável. A solução passa pela organização da chegada de produtos em lotes, de modo a evitar tempos de preparação e custos adicionais.

Esta solução, presente nas linhas multi modelo, obriga num curto espaço de tempo à solução de muitos problemas relacionados com a decisão de que lotes devem ser produzidos e qual a sequencia de montagem. No caso de um dos lotes ser demasiado grande, a abordagem poderá passar por uma solução separada para cada modelo, caso se verifique uma configuração entre lotes comparativamente pequena.

2.1.3. Dependênciado Controlo da Linha

A entrada de produtos numa determinada estação ou a saída do mesmo é da responsabilidade do controlo da linha, esse controlo pode tomar diferentes características consoante os objetivos e topologia do sistema (Boysen, Fliedner et al. 2008).

Numa linha cadenciada uma única passadeira transportadora de grandes dimensões é responsável por entregar todos os produtos. A passadeira transportadora mantem o seu funcionamento de uma forma constante, o que obriga a que cada operador efetue a sua tarefa até desde que o produto entra na sua área de trabalho até que este abandone a mesma. Como todas as estações são abastecidas pela mesma passadeira transportadora, a única alternativa é fazer parar cada produto nas estações um determinado tempo, igual para todos os produtos e estações, assim que este período termine cada produto é transportado até à próxima estação. Caso o tempo de execução numa das estações seja maior que o tempo de passagem do produto pela mesma, um tempo adicional de paragem deve ser previsto, de forma a possibilitar a correta execução da tarefa.

Uma solução assíncrona, linha sem cadência assíncrona, permite encaminhar cada produto para a estação seguinte, sempre que a tarefa a ser executada tenha sido terminada. Assim o produto nunca fica em espera nas estações, pois é prontamente despachado para a estação seguinte. Nesta abordagem o encaminhamento da peça é o mais rápido possível desde que a estação seguinte se encontre vazia e operacional. Cada decisão é tomada localmente por cada estação, de forma assíncrona, possibilitando o melhoramento dos tempos de espera em cada estação. De forma a contornar o problema da estação seguinte ocupada ou inoperacional, podem ser instalados *buffers* de forma a acumular os produtos já despachados. Desta forma as estações anteriores conseguem estar constantemente a operar mesmo que haja acumulação numa das estações posteriores.

Numa abordagem menos arrojada pode-se encontrar a linha sem cadência sincrona onde todas as peças presentes nas estações, apenas são mandadas avançar quando a estação mais lenta a operar terminar a sua tarefa. Assim sendo, quando todas as estações terminarem a execução da sua tarefa são avançadas todas as peças. Nesta implementação a utilização de *buffers* é desnecessária devido à sua natureza, em que todas as estações “esperam” pelo términoda estação mais lenta.

2.1.4. Instalação e Reconfiguração

A abordagem BL adequa-se particularmente a sistemas projetados de raiz e onde todo o horizonte de produção é antecipável o que é muito raramente o caso. Contudo é importante reter que após a configuração inicial e posterior instalação esta abordagem é de difícil reajuste e reconfiguração. Quando são necessárias alterações à produção muitas das variáveis utilizadas anteriormente deixam de estar disponíveis, pois muitas das vezes não há possibilidade de remodelar ou reposicionar estações, e mesmo que assim o seja todo o escalonamento e balanceamento têm de ser novamente adaptados a um dos casos anteriores e toda a configuração novamente feita. Esta abordagem tem como grande desvantagem a difícil reconfiguração e a falta de flexibilidade.

2.2. Sistemas Automatizados de Transporte

O encaminhamento automático de materiais é muito utilizado em algumas indústrias. Estes sistemas são frequentemente utilizados na recolha de matéria-prima e armazenamento da mesma, transferência automatizada de matéria-prima entre estações na fábrica, armazéns automáticos, transporte de contentores nos portos, manuseamento de bagagens nos aeroportos, entre outros.

Atualmente, o planeamento e controlo dos Sistemas Automatizados de Transporte (SAT) ou *Automated Material Handling Systems* na literatura são altamente personalizados e projetados para cada caso específico. A rigidez destes sistemas provoca inconvenientes que se refletem em aumento de custos e perda de desempenho.Um dos inconvenientes reside no facto de os requisitos ambientais e do sistema variarem ao longo do tempo, criando a necessidade de adaptação de procedimentos responsáveis pelo planeamento e controlo (Haneyah, Schutten et al. 2013). Devido às caraterísticas dos SAT, o tempo perdido e todo o trabalho necessário a uma reorganização e replaneamento do sistema obrigam a uma paragem na produção e a novos investimentos. Além disso, não recorrendo a estas alterações a linha torna-se ineficaz e obsoleta.

De forma a ultrapassar as limitações encontradas nos SAT estudos recentes têm sido realizados. Na literatura encontram-se estudos direcionados à resolução de situações muito específicas e outras direcionadas a resolver os problemas verificados nestes sistemas na generalidade.

Com o intuito de uniformizar e possibilitar a existência de um modelo de SAT comuma todos os sectores de mercado, em (Haneyah, Schutten et al. 2013) é proposta uma arquitetura que permite o controlo genérico de qualquer SAT. Esta abordagem propõe uma arquitetura responsável pelo planeamento e controlo da linha. Essa arquitetura é estruturada sobre a forma de *framework* onde as decisões são tomadas em várias camadas, onde as decisões em cada camada são influenciadas pela informação partilhada entre elas. A arquitetura proposta tem

como principal caraterística a flexibilidade, de modo a permitir uma fácil adaptação à mudança tanto de configuração como de objetivos, contendo diferentes modos. Esta abordagem permite que um administrador do sistema possa reconfigurá-lo ou reajustá-lo de forma a atingir novos objetivos.

Em (Hsieh, Cho et al. 2012), é proposta uma solução de SAT especifica para o encaminhamento de matéria-prima numa fábrica de semicondutores. Nesta proposta é considerada a existência de segmentos da célula compostos por dois caminhos bidirecionais, compondo um caminho com recirculação. Esta estrutura vem eliminar o congestionamento e o bloqueio de peças, sem ser necessário o recurso a investimentos adicionais pela operação dos VGA's por dois caminhos que não se intersectam. Esta solução permite uma melhoria significativa no ambiente onde está inserida, com o incremento da eficiência e a eliminação do congestionamento e interferências entre veículos, contudo é uma solução pouco flexível e pouco utilizável em outros ambientes de diferentes características.

Para facilitar a reconfiguração de um SAT, (Wong, Tan et al. 2007) propõem uma alternativa que se baseia em reconfiguração *on-line*. Esta solução aparece com o intuito de facilitar a reconfiguração do sistema em caso de falhas. Quando uma falha ocorre o sistema automaticamente verifica quais as rotas afetadas. Para as rotas afetadas pela falha são recalculadas novas rotas que possibilitem o mesmo encaminhamento, ficando as novas rotas em funcionamento. Esta solução permite uma recuperação dinâmica dos sistemas SAT em caso de falhas.

Regra geral os SAT permitem o encaminhamento de uma forma automática, com rotas estáticas e de flexibilidade praticamente nula. A solução proposta por (Wong, Tan et al. 2007) permite que o sistema seja dinamicamente alterado. Contudo este tipo de soluções implica uma estrutura mecatrónica já próxima da utilizada pelos modernos paradigmas de produção e tecnologias associadas.Embora esta solução seja uma mais-valia o problema do balanceamento de carga e otimização do encaminhamento continuam a ser descoradas.

2.3. Sistemas Flexíveis de Manufatura

Num Sistema Flexível de Manufatura (SFM) ou *Flexible Manufacturing System* na literatura cada ferramenta pode ser deslocalizada de uma máquina para outra. Esta habilidade de trocar ferramentas entre máquinas permite diferentes combinações entre máquinas e ferramentas, incrementando o número de operações possíveis. As máquinas presentes no sistema estão ligadas entre si por um sistema de encaminhamento de materiais, tanto este sistema como as máquinas são controlados por um sistema central.

Estes sistemas continuam a ser motivo de grande pesquisa e trabalho na busca de novas soluções. As pesquisas desenvolvidas num passado recente refletem-se tanto na

atribuição de execuções a conjuntos máquina-ferramenta como no encaminhamento do produto na linha.

Em (Jahromi and Tavakkoli-Moghaddam 2012) é proposto um modelo de modo a dinamizar a seleção do conjunto máquina-ferramenta e a respetiva alocação de operações a efetuar. Como se trata de um sistema SFM assumiu-se que existe um determinado conjunto de ferramentas e máquinas que possibilitam a execução de diferentes tarefas. O objetivo da solução proposta é fazer uma combinação de máquinas-ferramentas de modo a minimizar alguns custos de produção, sendo eles os custos de instalação, custos de manuseamento e custos de movimentação de ferramentas na linha. Contudo esta abordagem mantém uma natureza *NP-hard* tratando-se de um problema derivado do BL.

É proposta uma implementação de natureza distribuída em (Savkin and Somlo 2009), que executa um escalonamento em tempo real. Esta solução é proposta para sistemas fechados com recirculação. Para sistemas com estas caraterísticas, exclusivamente para estes, a abordagem proposta tem um comportamento convergente e estável. Com um comportamento deste género torna-se uma solução muito apelativa devido à consistência, não provocando problemas de imprevisibilidade.

De forma a introduzir uma nova área de estudo em sistemas desta natureza e de forma a melhorar o transporte de produtos e matéria-prima nos mesmos, (Sallez, Berger et al. 2009) propõem uma solução que engloba a vertente de transporte. É proposta uma arquitetura composta por duas camadas, uma física e uma virtual. Na camada virtual são constantemente calculadas as melhoras rotas para os produtos. Este cálculo é feito com o recurso ao algoritmo *Ant Colony* (Blum 2005) de uma forma acelerada, ou seja, na camada virtual o tempo é incrementado de forma a prever qual a melhor solução num determinado espaço de tempo. Nesta abordagem é também utilizado o conceito de produto ativo.

O relativo insucesso dos sistemas flexíveis deu origem a inúmeras abordagens que exploraram a natureza auto-organizada de sistemas baseados em componentes.

2.4. Sistemas Multiagente

Os Sistemas Multi Agente (SMA's) referenciados na literatura como *Multi Agent Systems* são constituídos por agentes autónomos que colaboram dinamicamente entre eles de forma a satisfazer tanto os objetivos locais como os objetivos globais. Tendo estas caraterísticas os SMA's são adaptados a ambientes de produção com o intuito de constituir ambientes compostos por diversos agentes heterogéneos, sendo eles do tipo ordem, trabalho a realizar, máquina, entre outros. Estes agentes têm diversos objetivos, restrições e comportamentos.

Este paradigma tem sido utilizado frequentemente para resolver problemas de escalonamento, não só no ramo da produção. Em (Kanaga and Valarmathi 2012) é apresentada

uma implementação desenvolvida em JADE (Bellifemine, Poggi et al. 1999) que permite o escalonamento de pacientes num hospital. Devido aos vários recursos disponíveis num hospital, o escalonamento de pacientes para a utilização de cada um deles levou a esta abordagem sustentada num SMA. Para encontrar o escalonamento ótimo recorreram à técnica meta-heurística, *Particle Swarm Optyimization*(Kennedy and Eberhart 1995).

A utilização de SMA's tem sido aplicada à produção maioritariamente na resolução do problema de escalonamento para cada um dos recursos disponíveis na linha. Algumas aplicações debruçaram-se sobre situações mais específicas como (Zhenqiang, Weiye et al. 2012). Neste caso é proposto uma solução para a otimização de linhas em que a topologia da rede, num determinado ponto aproxima-se à de um "gargalo", ou seja, existem vários pontos de entrada para um determinado recurso, fazendo com que a chegada de produtos a este recurso seja uma situação crítica de acumulação. Com esta solução o balanceamento entre a entrada de produtos num recurso deste género é controlada de foram a minimizar o congestionamento nos pontos que o antecedem, sendo assim o escalonamento para este fará com que os recursos anteriores mais preenchidos encaminhem para este mais rapidamente que os restantes descongestionando este ponto do sistema.

Numa aplicação puramente de escalonamento, onde as estações de serviço disponibilizam operações desempenhadas por humanos, (Sabar, Montreuil et al. 2009) propõem uma solução suportada por um SMA de forma a resolver algumas debilidades das soluções tradicionais que recorrem a *solve*. Nesta proposta os autores referem que em muitos dos casos a solução por *solver* é a que reflete um melhor desempenho do sistema, esta solução consiste na aproximação do sistema a um problema matemático e posteriormente resolvê-lo com uma programação matemática. A solução proposta prevê os casos onde é necessária uma rápida interação com o ambiente de modo a ter soluções num curto espaço de tempo. Nestes ambientes a utilização do *solver* fica impossibilitada.

Com o surgimento das linhas automatizadas foram desenvolvidos estudos com o objetivo de melhorar o escalonamento das máquinas que operam nessas linhas. Como já foi referenciado anteriormente esse escalonamento, em abordagens mais conservadoras pode ser feito pelo recurso a BL. A crescente utilização da informática aplicada à indústria despoletou a adaptação de inteligência artificial a estes sistemas, no controlo e otimização dos mesmos. Em (Lee, Choi et al. 2009; Guo and Zhang 2010) são propostas duas soluções aplicadas a sistemas de manufatura inteligentes, permitindo um escalonamento dinâmico e inteligente. Ambas as soluções são suportadas por uma plataforma multiagente de modo a que todas as entidades presentes no sistema colaborem num objetivo comum de realizar o melhor escalonamento possível de acordo com o algoritmo de escalonamento utilizado. Com soluções deste género os sistemas tornam-se mais flexíveis, refletindo-se numa redução do tempo de reconfiguração. Por sua vez esta caraterística reflete-se na diminuição dos custos de produção e do tempo de produção.

Em sistemas de produção é necessário fazer o planeamento e o escalonamento da produção. O planeamento é responsável por especificar quais os recursos presentes na linha e que operações são necessários para produzir um determinado produto. Quanto ao escalonamento é responsável por atribuir quais as operações que serão feitas por cada máquina presente na linha de maneira a que o plano de produção seja respeitado. Como é possível verificar, existe uma grande proximidade entre estas duas tarefas. Em (Li, Zhang et al. 2010) é apresentada uma solução que visa a integração entre as duas, de forma a melhorar o desempenho do sistema. A solução é suportada por uma arquitetura multiagente de forma a facilitar a integração e a otimização é baseada num algoritmo evolucionário.

Também com o objetivo de otimizar por meio da integração do planeamento e escalonamento, (Leung, Wong et al. 2010) propõem uma aplicação do algoritmo de Ótimização por Colónia de Formigas (OCF) (Blum 2005) baseada num sistema multiagente de modo a integrar o planeamento e escalonamento na linha de produção. De modo a recriar uma colónia de formigas, cada formiga é um agente independente. Cada formiga artificial naturalmente favorece o escalonamento que encontrar com menor tempo de produção total, deixando feromonas pelos caminhos por onde passa. Com as feromonas que as formigas artificiais vão deixando a probabilidade desse escalonamento ser o utilizado aumenta.

Em (Xiang and Lee 2008) propõem uma arquitetura suportada por um SMA, com recurso também a OCF de modo a melhorar a performance global do sistema. O objetivo desta abordagem é trabalhar unicamente no escalonamento dos recursos disponíveis. É proposto um escalonamento dinâmico, altamente reativo e autónomo.

Os VGA's têm sido vistos como recursos iguais aos restantes constituintes da linha, como tal nos sistemas inteligentes de manufatura é necessário que estes sejam inseridos nesse escalonamento. O facto de os VGA's serem vistos como recursos sem caraterísticas especiais faz com que as decisões tomadas por estas entidades sejam bastantes restritas, pois estes limitam-se a responder a pedidos efetuados por outras entidades, como o encaminhamento de um ponto para outro, previamente planeado para estas entidades.

Deste modo vários estudos têm vindo a ser desenvolvidos tendo em vista o desenvolvimento de arquiteturas de apoio ao escalonamento para estas entidades. Em (Erol, Sahin et al. 2012) é proposta uma arquitetura constituída por estas entidades e por todas as máquinas que constituem o resto do sistema. Esta arquitetura prevê um escalonamento para todas estas entidades, permitindo um alto nível de cooperação entre todos os elementos do sistema. Contudo a arquitetura apenas se debruça sobre a problemática do escalonamento.

Em (Giordani, Lujak et al. 2013) é proposta uma solução para VGA's unicamente. À semelhança de pontos referenciados anteriormente, nesta solução é trabalhada a integração entre o planeamento e o escalonamento de forma a incrementar o desempenho do sistema. A abrangência da arquitetura limita-se aos VGA's, não se refletindo nos restantes recursos.

2.4.1. Sistemas Holónicos de Manufatura

De forma a fazer frente às novas necessidades de produção, a customização em massa ou o baixo volume de encomendas, surgiram os Sistemas Holónicos de Manufatura (SHM) ou *Holonic Manufacturing Systems*. A ideia principal neste tipo de sistemas é proporcionar um processo de fabrico altamente dinâmico e descentralizado. Existem na literatura várias arquiteturas baseadas em holons nomeadamente PROSA (Van Brussel, Wyns et al. 1998), ADACOR (Leitão and Restivo 2006) e *Rockwell Automation Agents(Vrba, Tichy et al. 2011)*.

Os protótipos que se regem por este paradigma são habitualmente implementados sobre a forma de SMA. De forma a garantir o correto funcionamento do sistema, os sistemas de coordenação e controlo garantem que os planos são devidamente efetuados e que os prazos de execução de tarefas e de encaminhamento de produtos são cumpridos.

Baseando-se neste novo paradigma, (Vrba, Tichy et al. 2011) propõem um sistema desenvolvido para o controlo de sistemas de manufatura com base em equipamentos industriais. A arquitetura tem como principal ponto o facto de existirem agentes do tipo *Middle-Agent* responsáveis por formar equipas entre eles, disponibilizando tarefas de mais alto nível. Esta funcionalidade permite incrementar o desempenho do sistema devido à granularidade de algumas tarefas a serem efetuadas sobre o produto. Cada estação poderá ter uma única tarefa a ser executada por alguns recursos presentes nessa estação, mas do lado do produto é visto como sendo uma única tarefa, pois o *Middle-Agent* responsável por essa estação responsabiliza-se por formar uma equipa com esses recursos e coordenar as operações atómicas, efetuadas por cada um deles. Nas abordagens tradicionais existem duas camadas separadas, a LLC (*Low-Level Control subsystem*) que corre nos controladores e a HLC (*High-Level Control subsystem*) onde corre o agente, num computador separado. Nesta solução, os agentes desenvolvidos em C++ correm do lado do controlador sendo incorporados como uma camada superior da LLC. Assim o número de camadas é restringido a metade e a troca de informação simplificada, o que se reflete num melhoramento do desempenho do sistema.

2.4.2. Sistemas Biónicosde Manufatura

Os Sistemas Biónicos de Manufatura (SBM) ou *Bionic Manufacturing Systems* são inspirados no funcionamento dos órgãos naturais (Ueda 1992). Estes sistemas baseiam-se na hierarquia que é encontrada nos órgãos, havendo uma troca constante de informação entre as camadas que definem essa hierarquia, através de um caminho que regulamenta essa comunicação. A auto-organização é uma caraterística determinante neste paradigma, através da passagem de camada para camada da informação contendo o tipo de ADN. Esta informação permite reunir os submodelos que estão envolvidos na execução de cada tarefa. O objetivo é embutir num modelo principal as especificações de uma dada tarefa. Com esta informação este modelo principal analisa cada tarefa e seleciona os modelos de mais baixo nível de modo a reunir as condições para a execução da tarefa de mais alto nível.

Este paradigma tem sido frequentemente utilizado como base de começo para o desenvolvimento de novos SMA's, à semelhança do que acontece com os SHM.

2.5. Sistemas Reconfiguráveis de Manufatura

Os Sistemas Reconfiguráveis de Manufatura (SRM) ou *Reconfigurable Manufacturing Systems* na literatura embora sejam constituídos por inúmeras máquinas geridas por controladores industriais, máquinas responsáveis pelo transporte ou outro tipo de dispositivos, é nas reconfigurações máquinas-ferramentas que se encontra a caraterística mais notável neste paradigma de produção (Koren, Heisel et al. 1999).

As reconfigurações máquinas-ferramentas são o principal foco de desenvolvimento desta abordagem. Através de uma rápida personalização destas configurações é possível começar a produzir um produto desejado num curto espaço de tempo ou produzir uma nova mistura de produtos, nas quantidades pretendidas. As semelhanças com uma abordagem suportada por SFM terminam quando os SRM's facilitam a sua reconfiguração através da adição, remoção ou atualização de novas máquinas ou componentes. Com esta facilidade um SRM pode teoricamente, realizar uma expedita conversão entre produtos, dentro de uma determinada família de produtos, isto é, um SFM pode passar a produzir outro tipo de produtos caso uma reconfiguração assim o permita. Em (Galan, Racero et al. 2007) pode-se encontrar um pouco mais sobre este tipo de sistemas.

Alguns estudos e trabalhos têm sido desenvolvidos de modo a tornar a reconfiguração destes sistemas mais rápida e mais abrangente. Em (Li, Dai et al. 2009) é apresentada uma solução baseada em Redes de Petri. O controlo do sistema e da lógica de execução pode ser realizada por Redes de Petri e como tal nesta abordagem é apresentada uma solução que usa este recurso de modo a criar os modelos que regem o sistema. Neste caso apresenta-se uma solução em que são reconfigurados os modelos descritos pelas Redes de Petri de uma forma automática, sendo esta reformulação feita num nível mais baixo do habitual.

2.6. Sistemas Evolutivos de Produção

Sistemas Evolutivos de Produção (SEP's) ou *Evolvable Production Systems* (Onori, Alsterman et al. 2005) é um paradigma que surge com objetivo de projetar, manter e desenvolver sistemas industriais (Ribeiro, Barata et al. 2010). O SEP observa o sistema como um todo, em que o todo é constituído pelos produtos e por todos os outros elementos da linha de produção, sendo eles máquinas, ferramentas de configuração ou qualquer outro tipo de ferramenta.

O núcleo destes sistemas é constituído por entidades distribuídas proactivas e capazes de interagir entre elas, estas entidades podem ser dos mais variados tipos e ter vários objetivos.

Nestas entidades podemos ter produtos, tarefas, máquinas, ferramentas, habilidades, entre outros.

De forma a garantir o correto funcionamento destes sistemas, a sua integração é inspirada em sistemas biológicos, em particular nos conceitos de auto-organização e adaptação.

Estes sistemas têm na sua constituição uma interface inteligente que permite a capacidade de acoplamento ao sistema, assim este tem a capacidade de se auto-organizar caso seja requerido o acoplamento de novos elementos durante a sua execução. Esta capacidade tem impacto na forma como o sistema é desenhado e organizado e implica uma abordagem nova não só do controlo como detalhado na arquitetura mas também no diagnóstico (Ribeiro and Barata 2011).

A arquitetura SEP suporta o presente trabalho e em particular a arquitetura SMA discutida no capítulo da arquitetura é uma instanciação de SEP.

2.7. Conclusões Gerais

Como foi possível verificar, com a crescente mudança de exigências as abordagens tradicionais baseadas em BL tornaram-se obsoletas e inadequadas para os sistemas altamente dinâmicos apesar das suas inúmeras variâncias e alternativas. As abordagens tradicionais revelam-se demasiado estáticas e de difícil reconfiguração o que inviabiliza a sua utilização em sistemas onde esta exigência é uma constante.

Ao longo dos tempos várias abordagens foram propostas tendo em vista a resolução deste problema. O esforço inicial depositado nos SFM's acabou por se revelar um insucesso, o que levou ao aparecimento de inúmeros estudos e trabalhos baseados em sistemas auto-organizados.

Os SMA's tornaram-se objeto de grande estudo e desenvolvimento nesta problemática sendo frequentemente utilizados com o intuito de resolver o problema do escalonamento e planeamento ou em alguns dos casos ambos.

Mais recentemente foram apresentados os SEP's baseados em sistemas biológicos que apresentam uma solução evolutiva com o intuito de desenvolver uma solução altamente dinâmica e competitiva neste ambiente de novas exigências.

Com o estudo realizado verificou-se que frequentemente o sistema de transporte é excluído das tomadas de decisão relativamente ao balanceamento de cargas e otimização da produção, o que constitui uma enorme oportunidade de trabalho e estudo.

3. Arquitetura

Como justificado no capítulo anterior, a abordagem mais conservadora no escalonamento não lida bem com sistemas mais dinâmicos que potencialmente poderão ser gerados a partir do paradigma SEP. Neste contexto, esta arquitetura baseia-se em auto-organização e aparece como uma alternativa capaz de gerir estes sistemas mais dinâmicos.

A arquitetura proposta está inserida num contexto multiagente de modo a maximizar a capacidade do sistema. A autonomia conferida às entidades permite que estas reajam a mudanças durante a sua execução, introduzindo correçõesna sequência de processos de forma a uma distribuição de carga entre estações, de uma forma auto organizada.

Um ponto essencial do sistema é o *Product Agent* (PA). Este agente é capaz de decidir autonomamente, recorrendo ao apoio do Sistema de Transporte (ST) qual o melhor local para realizar o próximo passo do seu processo de produção. Com a ação coletiva entre ST, PA's e agentes capazes de disponibilizar habilidades cria-se uma dinâmica de auto-organização no sistema, suficiente para distribuir a carga entre os recursos existentes e assim reagir rapidamente à introdução ou remoção de outros recursos.

A abordagem proposta está inserida num ambiente constituído por oito tipos de agentes, sendo quatro delesreferentes ao ST. Os agentes responsáveis pelo transporte não fazem suposições relativas ao meio onde estão inseridos. O transporte é inteiramente administrado por estes agentes, tendo estes todas as capacidades necessárias à tomada de decisões relativamente ao encaminhamento do produto.

3.1. Arquitetura do Sistema

A arquitetura em estudo foi desenvolvida no âmbito do projeto IDEAS. A arquitetura do IDEAS (Figura 1) tem na sua constituição um núcleo, responsável pela execução do sistema. Para além do núcleo, a arquitetura do IDEAS é composta por ferramentas adicionais, permitindo a interação entre o operador humano e esse mesmo núcleo. O trabalho realizado incidiu sobre uma das parcelas constituintes do núcleo.

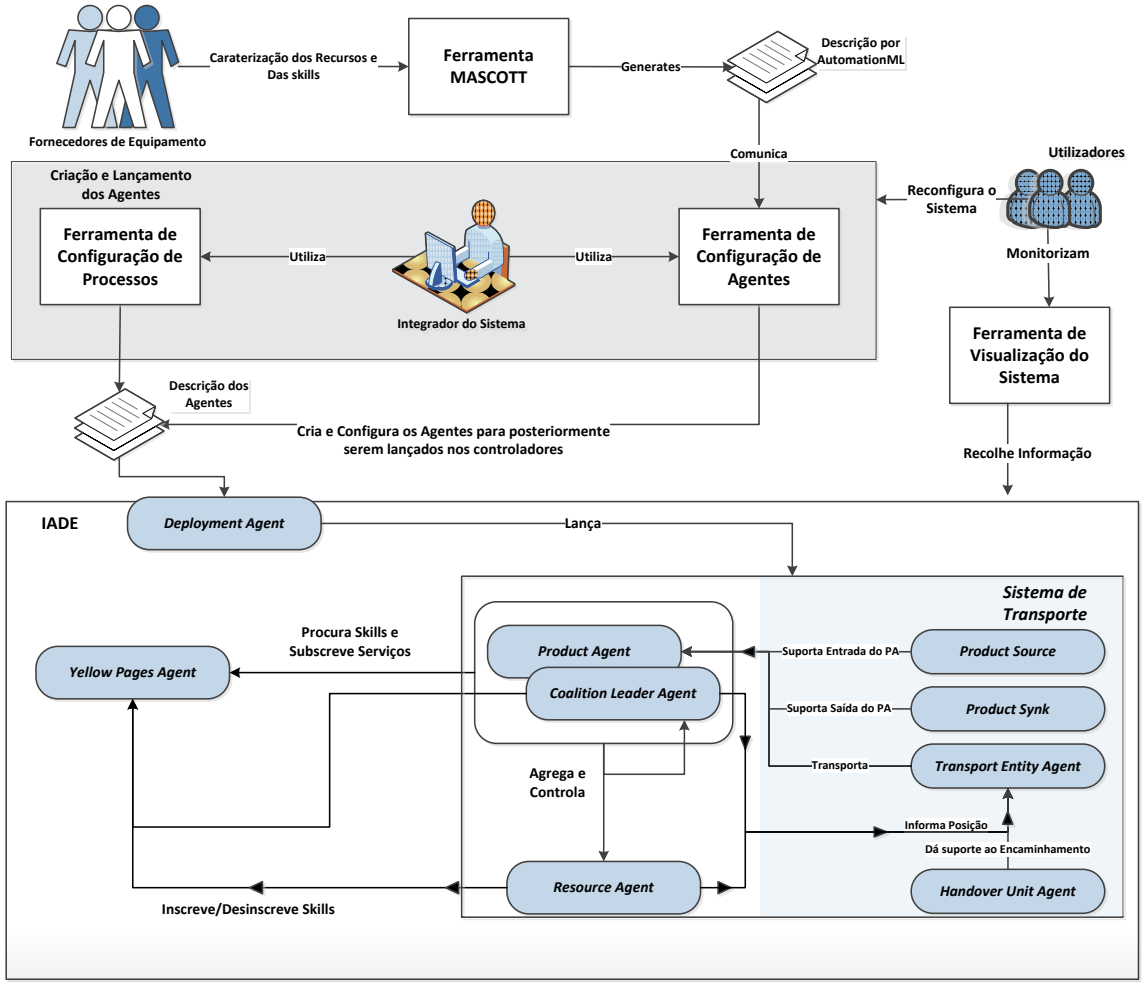


Figura 1 – Arquitetura do IDEAS

As ferramentas disponíveis, permitem ao utilizador manipular e consultar a execução do sistema. A ferramenta de configuração de agentes desenvolvida pelo UNINOVA permite ao utilizador configurar todos os agentes que serão posteriormente lançados nos controladores, permitindo definir que recursos estarão disponíveis e onde, cada elemento do ST que vizinhos tem e que caraterísticas, entre outras possibilidades. A ferramenta MASCOTT desenvolvida pela MASMEC, utilizada por quem disponibiliza o *hardware* dos recursos, é utilizada para descrever todas as tarefas que o mesmo pode fazer, como os movimentos do robô. Estas descrições são apresentadas num ficheiro AML. A construção de processos, para serem usados como plano de execução de um PA ou disponibilizados por um CLA, é realizada recorrendo à ferramenta de configuração de processos desenvolvida pela Universidade de Nottingham. Por fim o utilizador pode, em tempo real, visualizar o que está a ser realizado pelo sistema, como em que estação está um PA e que habilidade está a ser executada, e por que RA. Também está disponível na ferramenta de visualização desenvolvida pela KTH Royal Institute of Technology, um historial onde é possível consultar que habilidades já foram executadas em cada PA, e o resultado das mesmas.

A arquitetura do IADE, o núcleo responsável pela execução do sistema, é composto por um total de nove agentes. Estes agentes estão divididos de acordo com o seu papel no sistema. Os agentes relevantes para o trabalho em estudo são os agentes diretamente relacionados com o transporte, representados na Figura 1 no espaço a azul. De um ponto de vista funcional, o papel de cada agente é descrito na Tabela 1.

Tabela 1 - Agentes Mecatrónicos e Funções

Grupo	Classe	Função
Suporte	***DeploymentAgent*** **(DA)**	**Medeia a plataforma de agentes e os controladores da linha. Permite ao utilizador lançar agentes responsáveis pela execução ou transporte num determinado controlador.**
	YellowPagesAgent **(YPA)**	**Mantem o controlo dos serviços/habilidades que cada um dos agentes disponibiliza no Sistema. Os agentes podem consultar o YPA para localizar outros agentes e as suas habilidades. Cada agente pode assinar um serviço de notificação que o informa, se um agente específico deixou a plataforma.**
Execução	***Resource Agent*** **(RA)**	**Interagem diretamente com os controladores da linhae representam a entidade de menor abstração possível da pilha do IADE. O seu principal papel é o de traduzir habilidades para código nativo e assegurar a sua execução e sincronização com a plataforma de agentes. Os RA's informam o ST sobre a localização física onde as suas habilidades podem ser executadas.**
	CoalitionLeaderAgent **(CLA)**	**Os CLA's são utilizados para compor funcionalidades recorrendo a agentes já existentes. Não interferem com qualquer controlador da linha, mas são capazes de coordenar a execução de outro RA ou CLA. Estes implementam subprocessos que serão posteriormente consumidos por agentes do tipo produto, durante a execução do seu plano de processo. À semelhança do RA, o CLA informa o ST sobre a localização física onde as habilidades estão disponíveis.**
	ProductAgent **(PA)**	**O PA representa o nível mais alto da abstração do sistema e que implica um mapeamento direto entre o agente e um item específico a ser produzido. Cada PA tem a capacidade de controlar o seu próprio plano de execução e de tomar decisões sobre o local onde cada passo dessa execução é executado. A ação coletiva do PA e os agentes do ST permite a este basear as suas decisões.**
Transporte	***ProductSourceAgent*** **(PSoA)**	**O PSoA gere a entrada de paletes no sistema e posteriormente, quando esta já se encontra no sistema, associa-lhe um PA específico. Esta associação é fundamental, já que pode haver mais que um PA disponível para entrar, e neste contexto, o PSoA gere a entrada de cada PA em espera.**
	ProductSinkAgent **(PSiA)**	**O PSiA tem a função oposta ao PSoA. Neste contexto, o PSiA liberta as paletes do PA, sempre que um PA especifica termina o seu plano de execução.**
	TransportEntityAgent **(TEA)**	**Abstrai um transportador específico do sistema. É responsável pelo deslocamento de paletes entre encaminhadores. Cada TEA disponibiliza vários pontos de acoplamento, onde as estações, constituídas por RA's e CLA's, podem ser acopladas e desacopladas durante a execução. O TEA é sensível a essas ações e gere o tráfego das paletes no mesmo, informando o PA sobre o custo do encaminhamento e a posição atual aquando da chegada ao destino.**
	HandoverUnitAgent **(HUA)**	**Cada HUA controla um encaminhador do sistema e calculam a árvore de custos mínimos para chegar a cada um dos destinos possíveis na linha. Esta informação é posteriormente partilhada como TEA associado, para quando este necessita negociar com o PA.**

O controlo do ST do sistema em estudo, é da responsabilidade de quatro agentes, todos eles com procedimentos exclusivamente relacionados ao encaminhamento do produto, onde incidiu o trabalho realizado.

De forma a facilitar a compreensão do documento serão consideradas estações os pontos de acoplamento onde estejam alocados RA's, CLA's ou agentes dos dois tipos. Também é importante referir que numa visão física do sistema encaminham-se paletes, mas estas são abstraídas pelos agentes do tipo PA.

3.2. Noção de Habilidade

Os agentes presentes no ambiente em estudo expressam as suas funcionalidades em habilidades. As habilidades, inicialmente definidas em (Barata 2005), sobre a forma de *skill*, desempenham para o agente o mesmo papel que um método para um objeto. Sendo assim, uma habilidade inclui uma interface que contém toda a informação necessária à sua execução, requerida por um produto. Nessa informação encontram-se:

- Nome e ID;
- Parâmetros e os seus tipos;
- Referência ao agente que a irá executar;
- Variáveis de estado, que permitem o controlo de execução;

O tipo de habilidade, é fundamental durante o processo de execução e pode assumir um dos seguintes valores:

- Habilidade Atómica (HA) / *Atomic Skill*: Implementam mecanismos genéricos que associam a informação da *skill* à execução do mesmo num controlador específico (procedimentos de baixo nível, programas e outros sistemas de integração);
- Habilidade Complexa (HC) / *Complex Skill*: Implementam processos. Recorrendo a um *work-flow* como descritor, estes constituem novas *skills*. Também é definido se estes são executados em paralelo, sequencialmente, e também se existem instruções condicionais que afetam o fluxo normal de execução;

3.3. Arquitetura do Sistema de Transporte

A arquitetura do ST proposta é constituída por um sistema de passadeiras transportadoras e encaminhadores. Como podemos ver na Figura 2as passadeiras transportadoras funcionam como elos de ligação entre encaminhadores sendo estes os pontos

de redirecionamento e encaminhamento das paletes. Nos encaminhadores são tomadas as decisões referentes ao trajeto realizado pelo produto, durante o seu transporte, doravante chamar-se-á simplesmente passadeiras quando se quiser referir a passadeiras transportadoras.

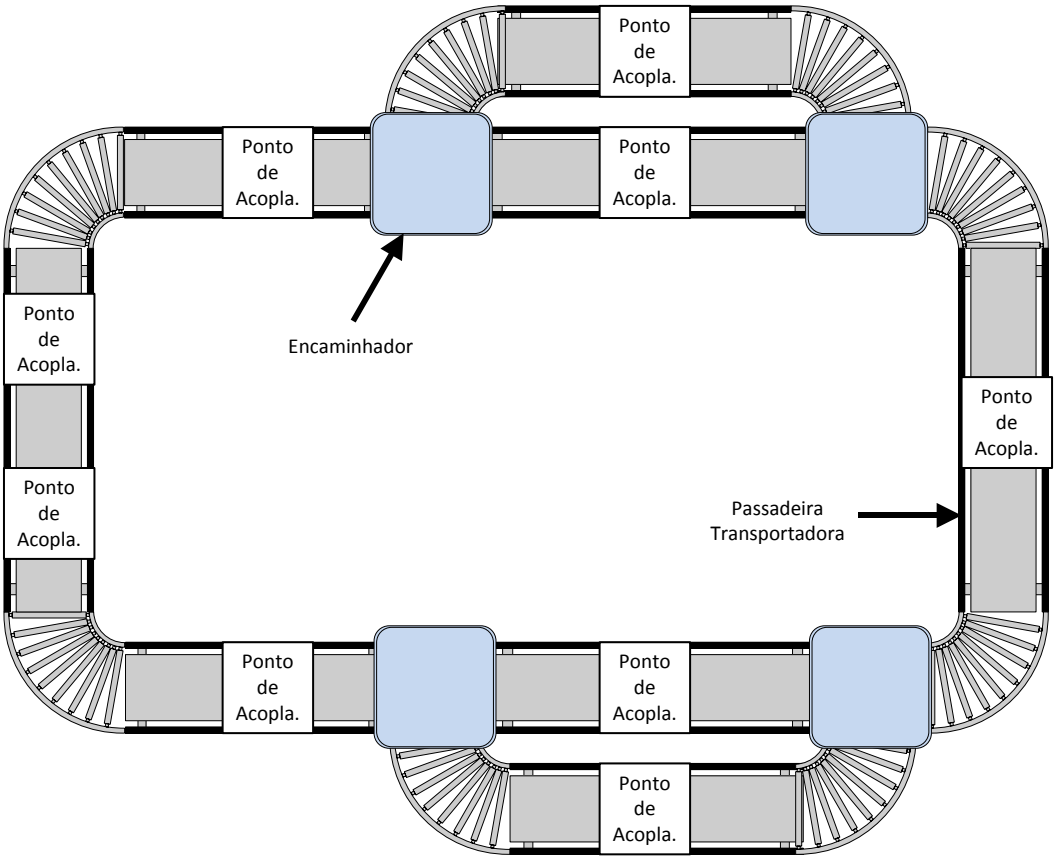


Figura 2 – Esquema representativo do sistema de transporte estudado

Inicialmente, na conceptualização da arquitetura, foram considerados os quatro agentes, já anteriormente referenciados, sendo estes o *Transport Entity Agent*(TEA), o *Handover Unit Agent* (HUA), o *Product Source Agent* (PSoA) e o *Product Sink Agent* (PSiA).

Das quatro entidades propostas, aquelas que têm tarefas com características mais aproximadas, são naturalmente o PSoA e o PSiA. O PSoA é responsável pela introdução de um agente do tipo PA no ambiente de produção, sendo esta introdução, tanto a nível físico como a nível lógico. Esta entidade comporta-se como um HUA de características especiais. Estes agentes são responsáveis pela primeira decisão tomada pelo ST. Quanto ao PSiA, tem como função receber os PA's, quando estes chegam ao fim do ciclo de produção, resumidamente, quando estes já executaram todas as habilidades que pretendiam no sistema. À chegada do produto a este HUA, também de características especiais, o PSiA informa o PA que este terminou a sua execução, fazendo com que este seja removido do ST, por sua vez o PA termina automaticamente a sua execução removendo-se do sistema, pois já não há interesse para o sistema que este agente continue a correr.

No ST facilmente se percebe que os agentes de maior preponderância são, os responsáveis por abstrair as passadeirase os encaminhadores, sendo eles o TEA e o HUA respectivamente.

De modo a possibilitar a associação de recursos ao sistema, que disponibilizam as habilidades requeridas pelo PA, as passadeiras permitem o acoplamento de estações ao longo das mesmas. Estes pontos chamados de pontos de acoplamento, permitem ao utilizador acoplar fisicamente o *hardware* à passadeira e também que os RA's e/ou CLA's presentes nessa estação se associem ao TEA que abstrai essa passadeira.

3.3.1. Representação Lógica

Do ponto de vista lógico o sistema é abstraído por um grafo direcionado. No grafo resultante os encaminhadores são representados como os nós do grafo e as passadeiras como os ramos constituintes do mesmo.

Pegando no exemplo exposto naFigura 3, o grafo resultante desta representação lógica tem o seguinte aspecto.

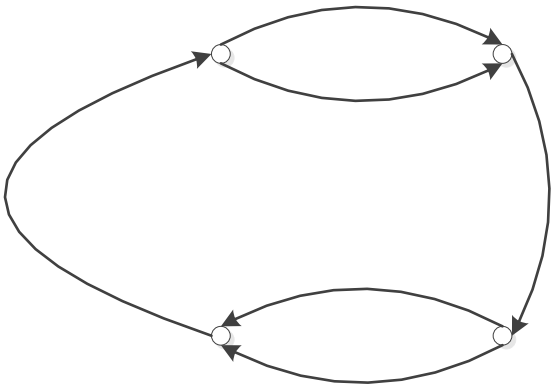


Figura 3 – Grafo representativo do sistema

De notar também que o grafo tem de ser unidirecional, pois as passadeiras encaminham apenas numa direção. Como se pode verificar, a representação sobre a forma de grafo, neste caso específico, é bem representativo do sistema a correr. O custo de percorrer cada um dos ramos do grafo vai depender de algumas variáveis em estudo, possibilitando, como demonstrado mais tarde o balanceamento da carga, de forma auto-organizada, por ajuste destas variáveis do sistema.

3.3.2. Cálculo do "Caminho mais Curto"

O objetivo de um ST não passa única e exclusivamente por fazer chegar o produto desde a sua posição atual até ao ponto desejado, ou seja, a estação onde este poderá requerer a execução da habilidade pretendida.

Para além deste objetivo inicial, é também interessante o conceito de que este o fará de uma forma eficiente e rápida, tendo em conta esta necessidade o caminho escolhido é sempre o caminho com menor custo, de acordo com a métrica utilizada.

Sabendo-se de antemão os pesos associados, aos ramos do grafo, é necessário calcular o caminho mais curto, desde o nó em que o produto se encontra até cada um dos pontos alcançáveis do sistema. De forma a tornar possível esse cálculo recorreu-se ao Algoritmo de *Dijkstra* (AD) *(Tanenbaum 1996; Gou, Luh et al. 1998)*.

3.3.2.1. Algoritmo de *Dijkstra*

Este algoritmo foi desenvolvido por *Edsger Dijkstra* em 1959, com o intuito de resolver a problemática do caminho mais curto num grafo, seja ele direcional ou não direcional, em que a única restrição é o facto de as arestas não poderem ter pesos negativos. Este algoritmo tem complexidade $O([m+n]\log n)$, sendo m o número de ramos e n o número de nós que constituem o grafo.

O Algoritmo de *Dijkstra* (AD) toma a decisão que é ótima no momento em que é executado, sem ter em conta a história do sistema ou qualquer tipo de projeção, relativamente a um comportamento futuro do mesmo.

Sendo calculado um caminho entre dois nós do grafo, rapidamente se percebe que todo o subcaminho, que constitui o caminho na sua totalidade, é um caminho de custo mínimo entre esses dois nós, esta característica faz com que o algoritmo seja baseado na construção dos melhores caminhos dos nós alcançáveis pelo nó inicial, determinando todos os caminhos intermédios.

O algoritmo tem como base uma tabela, que é utilizada para guardar os nós já visitados e, se estes já foram considerados permanentes ou não, um nó ser considerado permanente ou não durante a execução do algoritmo,depende do facto de o caminho já calculado ser o definitivo, ou seja, que aquele valor e aquele caminho são o caminho mais curto para aquele destino.

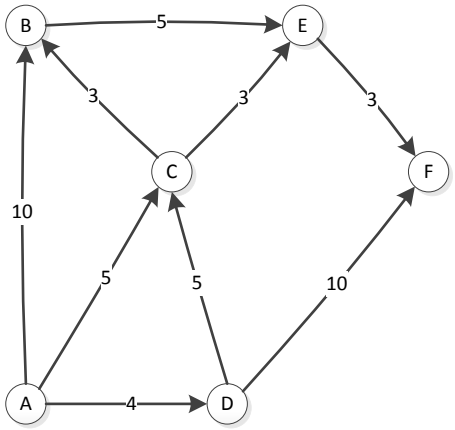


Figura 4 – Exemplo de um grafo completo

Tendo como exemplo o grafo da Figura 4, e objetivando-se o cálculo da árvore de caminhos mais curtos para o nó A, o primeiro passo no cálculo da árvore é marcar o nó de onde são calculadas todas as distâncias e rotas como raiz de uma árvore, e marcá-lo como nó permanente.

Sabendo todos os nós constituintes do grafo, a tabela de custos mínimos é inicializada com 0 no custo de atingir a raiz e infinito para todos os outros nós, e os precedentes para cada nó são inicializados com um nó por defeito. Estes dados são atualizados à medida que se avança no cálculo da árvore.

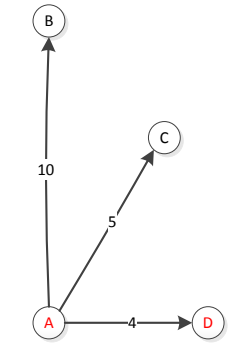
Posteriormente em cada iteração, sendo um nó '*X'* o próximo da lista a tornar permanente por ser aquele com menor custo, aquele que será tornado permanente. Depois a partir deste nó '*X',* calcula-se a soma do custo de alcançar um vizinhoe o de alcançar '*X'*, caso esta soma seja menor que o custo guardado na tabela de custo, a entrada é substituída com o precedente '*X',*e o custo anteriormente calculado, ou seja, o custo de alcançar '*X'* mais o custo de '*X'* alcançar esse vizinho.O critério de paragem do algoritmo resume-se à totalidade dos nós já terem sido atribuídos como permanentes.

Na Tabela 2 pode-se ver os passos necessários ao cálculo da árvore de custos mínimos aplicando o AD ao grafo da Figura 4, com o nó A a ser a raiz da mesma.

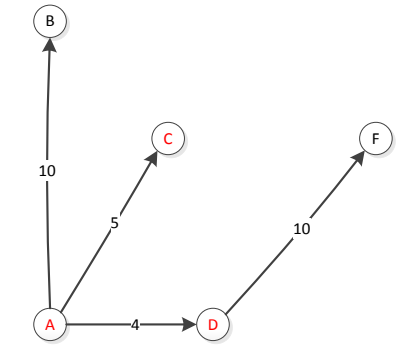
Tabela 2 – Cálculo de Algoritmo de Dijkstra

Árvore de Custos mínimos	Tabela de Custos

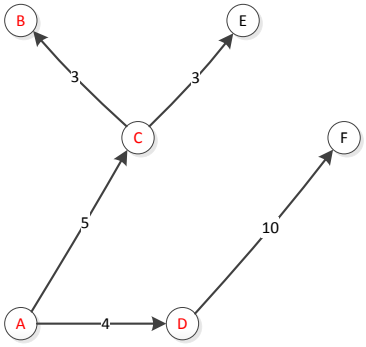
Nó	Custo	Prec.	Cam.	Perm.
A	0	-	-	Sim
B	∞	-	-	Não
C	∞	-	-	Não
D	∞	-	-	Não
E	∞	-	-	Não
F	∞	-	-	Não



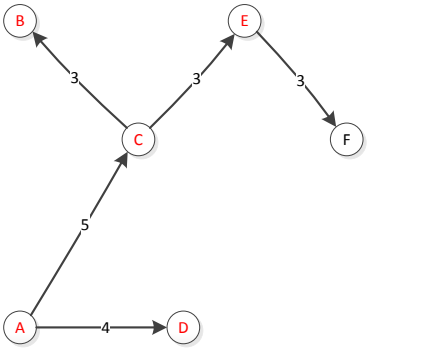
Nó	Custo	Prec.	Cam.	Perm.
A	0	-	-	Sim
B	10	A	B	Não
C	5	A	C	Não
D	4	A	D	Sim
E	∞	-	-	Não
F	∞	-	-	Não



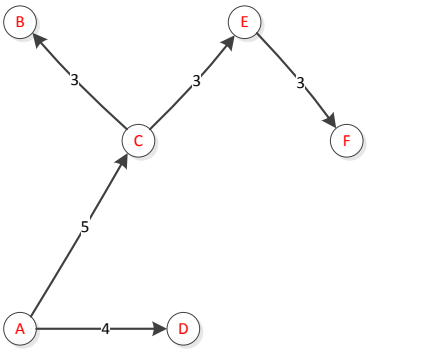
Nó	Custo	Prec.	Cam.	Perm.
A	0	-	-	Sim
B	10	A	B	Não
C	5	A	C	Sim
D	4	A	D	Sim
E	∞	-	-	Não
F	14	D	D	Não



Nó	Custo	Prec.	Cam.	Perm.
A	0	-	-	Sim
B	8	A	C	Sim
C	5	A	C	Sim
D	4	A	D	Sim
E	8	C	C	Não
F	14	D	D	Não



Nó	Custo	Prec.	Cam.	Perm.
A	0	-	-	Sim
B	8	A	C	Sim
C	5	A	C	Sim
D	4	A	D	Sim
E	8	C	C	Sim
F	14	D	D	Não



Nó	Custo	Prec.	Cam.	Perm.
A	0	-	-	Sim
B	8	A	C	Sim
C	5	A	C	Sim
D	4	A	D	Sim
E	8	C	C	Sim
F	11	E	C	Sim

No caso da arquitetura desenvolvida os HUA's são representados no grafo pelos nós e os TEA's pelos ramos do mesmo. No exemplo da Tabela 2, a árvore de custos mínimos calculada representa os melhores caminhos do HUA com o nome A até todos os restantes HUA's.

3.4. *Handover Unit Agent*

Numa linha de produção, frequentemente são necessárias tomadas de decisão quanto à próxima passadeira por onde deverá circular o produto ou material. Para além dessa componente é preciso ter entidades mecânicas capazes de encaminhar fisicamente o produto, refletindo as decisões efetuadas na camada lógica.

Tanto essas entidades mecânicas, neste caso encaminhadores, como toda a lógica por detrás do seu controlo, são da responsabilidade do HUA, possibilitando uma eficiente e correcta decisão quanto ao encaminhamento do produto, em cada ponto de decisão, efetuando as ações necessárias para que essa decisão se reflita no mundo real, acionando o encaminhador que abstrai.

3.4.1. Habilidade de Conhecer o Sistema como um Todo

De forma a tornar possível a tomada de decisões, tendo em vista o encaminhamento do produto pelo melhor caminho, o HUA necessita de uma visão global do ST. Esta visão é conseguida pela aprendizagem de todos os HUA's existentes no sistema, os seus vizinhos (dois TEA's) e os HUA's que são alcançáveis recorrendo a esses mesmos vizinhos.

Para que esta informação seja trocada entre os HUA's, estes partilham-na entre eles, possibilitando assim a criação de uma visão global do sistema, para que cada um construa o grafo representativo do sistema.

São necessários três comportamentos, resultando em dois tipos de interações descritas na Figura 5, que tornam exequível a aquisição desta informação, pois é necessária a partilha de informação entre o HUA e os TEA's vizinhos e posteriormente com todos os outros HUA's presentes no sistema.

Para além da informação que já foi dita, é também necessário conhecer o custo de percorrer uma passadeira abstraída pelo TEA, que torna os HUA's alcançáveis. Quando o HUA pede aos TEA's seus vizinhos, os HUA's alcançáveis, este responde com o identificador do HUA e o custo de o alcançar.

Consequentemente, quando o HUA for requisitado por outro, tendo em vista a renovação da informação, este já poderá enviar, não só os HUA's que podem ser alcançados, como também o custo de percorrer o ramo da responsabilidade deste TEA, que fará chegar o produto até esse HUA.

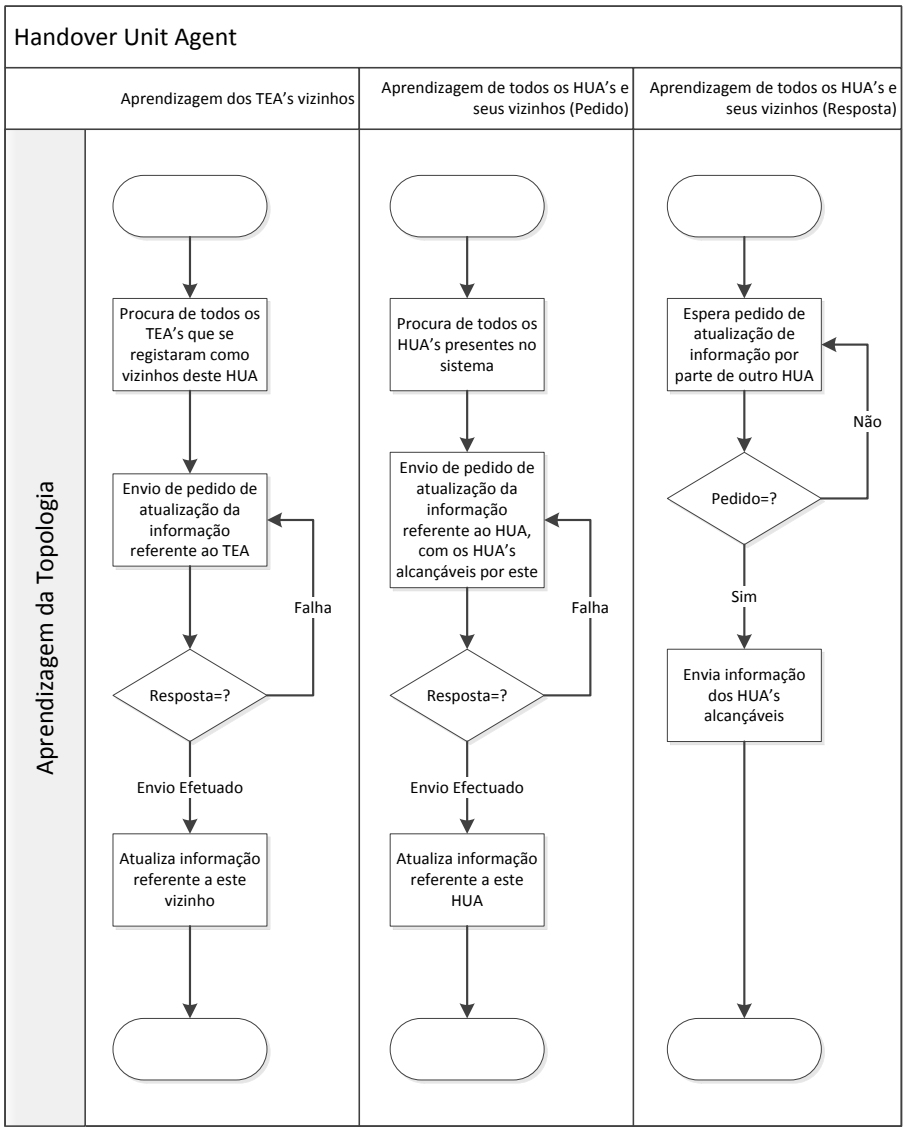


Figura 5 - Interacções necessárias à aprendizagem da topologia do sistema

3.4.2. Cálculo e Atualização da Tabela de Encaminhamento

O principal objetivo do HUA, como já foi referido é o de "descobrir" qual o melhor caminho para atingir um destino, e assim poder encaminhar o produto para o próximo TEA, quando este se encontra no seu espaço de ação.

Cada HUA tem uma representação lógica da topologia da rede, que constituirá o grafo de todo o ST. Apenas a construção do grafo não é suficiente para um encaminhamento eficiente, pois conhecem-se todas as opções de encaminhamento mas não a melhor opção, como tal recorre-se ao AD para calcular, sempre que se justifique, a Tabela de Encaminhamento (TE).

A TE constitui a base de decisão do HUA, referente ao encaminhamento, sendo assim, cada nó de destino tem uma entrada na tabela, em que é indicado qual o próximo nó, sendo este nó alcançável pelo atual.

Para tornar possível a construção e atualização desta tabela recorre-se a uma rotina que, sempre que a informação recebida referente à topologia seja atualizada, é reiniciadade forma a que a TE seja atualizada. Caso uma atualização ainda esteja a decorrer, espera-se que esta termine e posteriormente inicia-se a nova atualização, resultando no comportamento representado pela Figura 6.

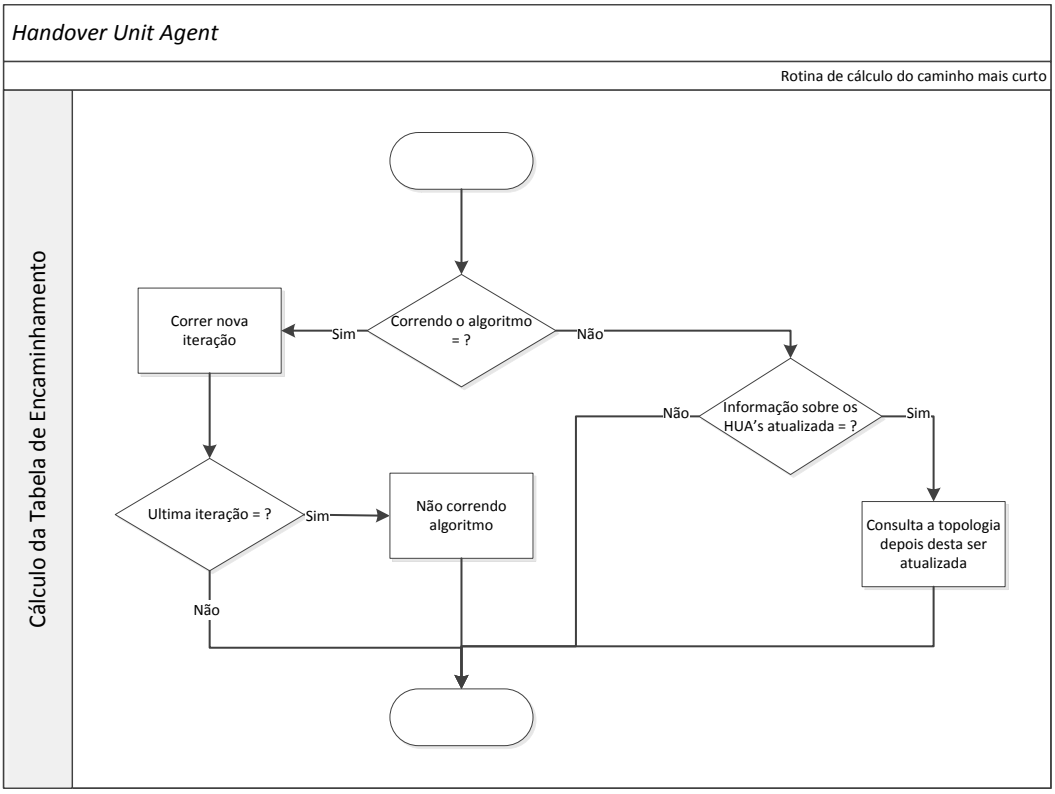


Figura 6 - Rotina responsável pelo cálculo da TE

3.4.3. Conhecer as Habilidades Alcançáveis por cada HUA

No encaminhamento do produto o ST inicialmente envia-o para o TEA onde se encontra acoplada a estação de destino. Após a chegada a esse TEA, este comunica ao PA a sua chegada à estação, para que este posteriormente peça a execução da habilidade pretendida.

Para que tal seja possível, é preciso que cada HUA tenha presente, na informação sobre cada um dos TEA's de saída, que habilidades e estações estão acopladas naquela mesma passadeira, abstraída por esse TEA, e assim essas habilidades serem consideradas alcançáveis por esse HUA. Resulta então, que este é o último a ser percorrido antes de se atingir a estação com a capacidade de executar essa habilidade, isto é importante no que respeita ao encaminhamento, visto que é o nó do grafo para onde é necessário encaminhar o produto. Quando esse destino é alcançado, o HUA responsável verifica que é uma habilidade alcançável por este e encaminha para o respectivo TEA.

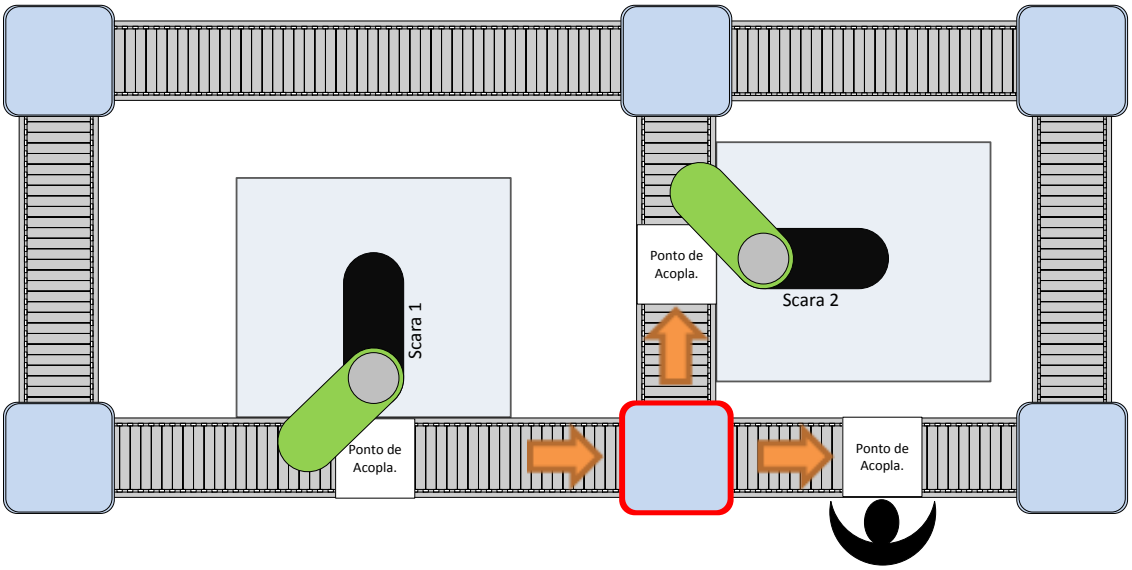


Figura 7 – Célula representativa da atribuição de habilidades a um HUA

Na Figura 7 as setas a cor-de-laranja representam o sentido de deslocamento das passadeiras à entrada e à saída do encaminhador delimitado a vermelho.

O encaminhadordelimitado a vermelho tem à sua saída duas passadeiras, fazendo com que lhe estejam atribuídas as habilidades associadas a essas passadeiras, neste caso as habilidades operadas pelo *Scara* 2 e as da responsabilidade do operador humano. As habilidades operadas pelo *Scara* 1, não poderão ser atribuídas a este HUA, pois a passadeira não é a de saída mas sim a de entrada. Neste caso o HUA responsável por estas habilidades é o que o precede, sendo neste caso o que se encontra à sua esquerda.

3.4.4. Entrada e Saída do Produto no e do Encaminhador

Outra das funcionalidades do HUA necessária ao correto funcionamento do encaminhador, é a capacidade de gerir a entrada e saída de produtos dele mesmo.

No modelo desenhado, assumiu-se que cada encaminhador tem, única e exclusivamente, capacidade para ter em sua posse um produto, resultando em que, cada entidade destas só poderá receber um produto quando este tiver “despachado” o anterior para a respetiva passadeirade destino.

Na Figura 8 está representado um exemplo de um encaminhador, com duas passadeiras de entrada, em que o sentido do deslocamento está assinalado pelas setas vermelhas, e duas de saída, onde as setas com a mesma finalidade estão a verde.

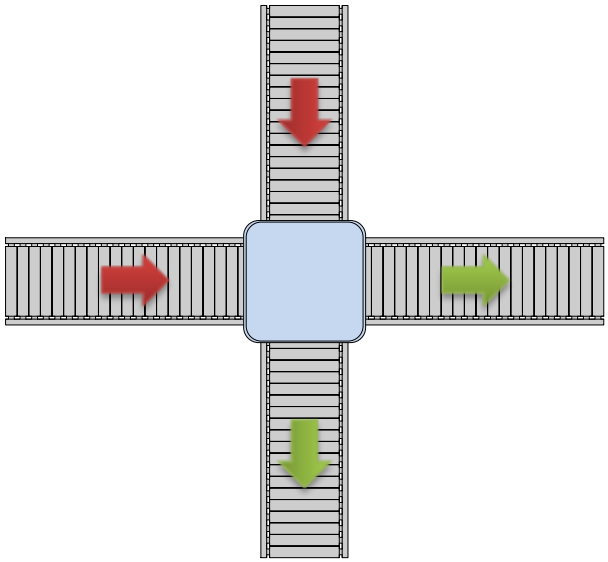


Figura 8 – Exemplo de encaminhador com duas passadeiras de entrada e duas de saída

Existem alguns casos delicados que necessitam ser tratados, de forma a não gerar conflitos de acesso a uma mesma posição por parte dos produtos, e de controlo de fluxo dos mesmos.

Caso o encaminhador esteja desocupado, o HUA deverá aceitar a entrada de paletes provenientes de uma das passadeiras à entrada, respondendo afirmativamente ao pedido feito pelo TEA. Quando uma palete se encontra no encaminhador, o HUA verifica o destino do PA, caso este seja um dos seus TEA's de saída envia-o para o vizinho. Caso a situação anterior não se verifique o HUA consulta a TE de modo a saber qual o próximo HUA de destino, de modo ao PA atingir o seu destino.

Na Figura 9,pode-se consultar o esquema que representa o comportamento do HUA, permitindo a correta chegada de produtos aos encaminhadores e seguinte despacho.

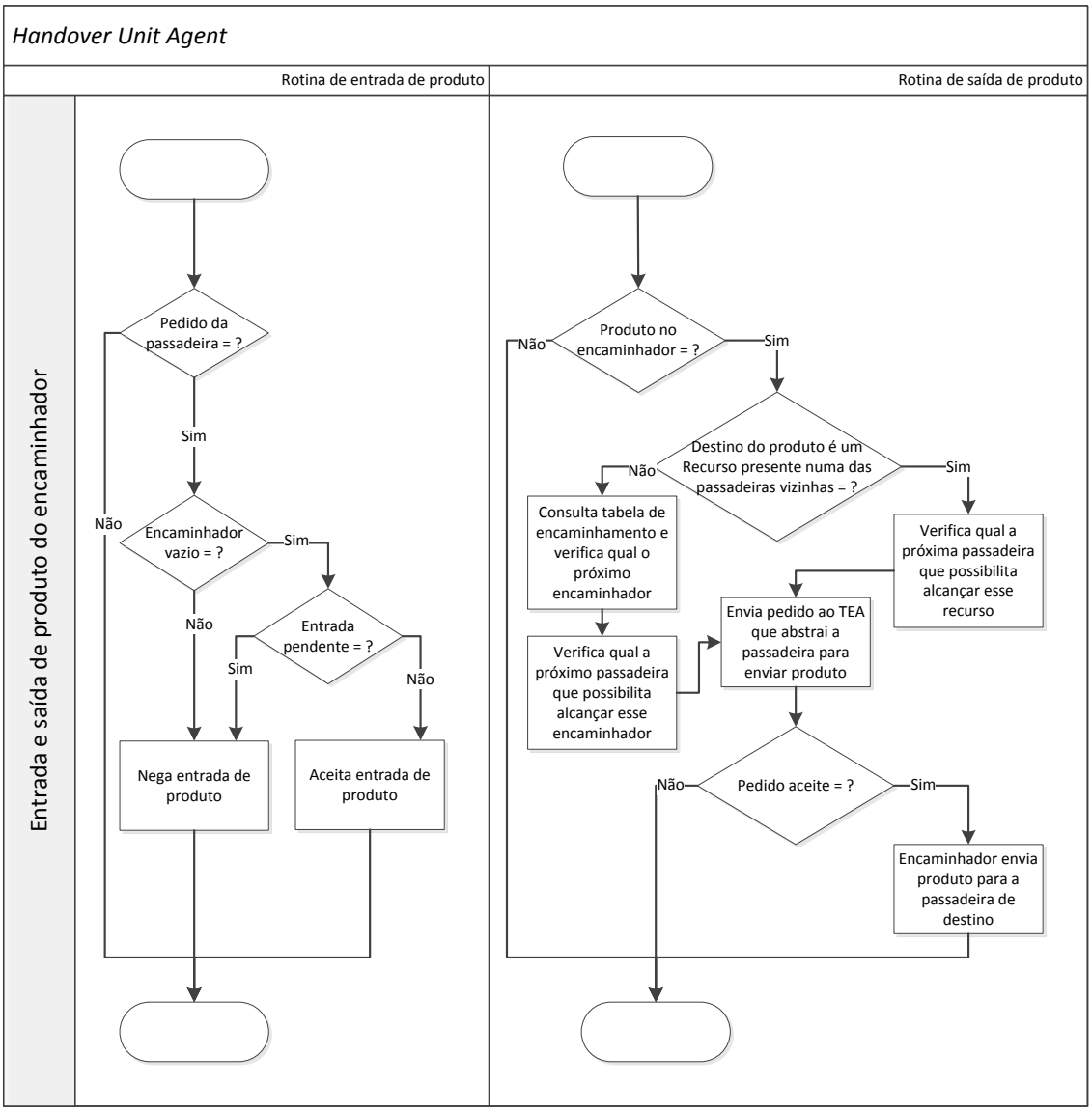


Figura 9 – Esquema com as rotinas de entrada e saída de produto no encaminhador

A rotina responsável pela entrada de produtos no encaminhador é despoletada quando um TEA vizinho inicia um pedido de encaminhamento de produto. Caso o encaminhador abstraído por este HUA se encontre vazio ou este não tenha negociado já uma entrada a resposta é positiva.

A rotina de despacho do produto do encaminhador é realizada de forma cíclica, permitindo que frequentemente o HUA verifique a existência de um produto no encaminhador. Caso exista e este ainda não tenha sido processado, o HUA consulta o destino deste e inicia a negociação com o TEA para onde tenciona encaminhá-lo.

3.4.5. Acoplamento e Desacoplamento de estações

Quando é realizado o desacoplamentode estações num ponto de acoplamento da passadeira, funcionalidade possível e descrita na arquitetura do TEA, o TEA responsável por essa mesma passadeira comunica ao seu HUA de entrada que os RA's e/ou CLA's alocados a essa estação deixam de estar disponíveis. Esta comunicação é importante para o caso, já anteriormente referido, de cada HUA saber quais os RA's e CLA's alcançáveis.

Num caminho inverso, e porque o ST arranca sem qualquer estação acoplada, cada TEA onde é feito o acoplamento de uma estação deverá comunicar ao HUA vizinho responsável por lhe dar entrada de produtos, que estão disponíveis esses novos RA's e/ou CLA's.

Esta habilidade tem como objetivo facilitar e tornar flexível o ST, para que este possa ser reconfigurado, ao nível das estações disponíveis, em qualquer altura da sua execução.

3.4.6. Adicionar e Remover Passadeiras Transportadoras ao Sistema

O ST suporta a adição e remoção, em qualquer período da sua execução, de passadeiras. No que respeita aos HUA's, estes recebem o pedido para que seja adicionado como vizinho do TEA, caso a passadeira seja adicionada. Pelo contrário, quando uma passadeira deseja abandonar o sistema ou simplesmente ser desativada, o TEA deverá comunicar a sua vontade aos HUA's vizinhos, quando estes receberam a resposta, e sendo esta afirmativa, termina então a execução do TEA que a abstrai. Os HUA's que recebam este pedido, ou seja, vizinhos dessa passadeira, deverão verificar se este TEA é responsável, por uma passadeira de saída, se assim for este deverá apagar dos seus dados o HUA que era alcançável por essa mesma passadeira e consequentemente também apagar os dados referentes às estações anteriormente acopladas a essa passadeira.

De forma a que toda a topologia da rede seja conhecida, por cada um dos HUA's, o HUA de entrada que recebe o pedido de remoção da passadeira passa a não conter nos seus vizionhos o HUA alcançado através desta passadeira. Este compurtamento faz com que este ramo do grafo passe a não existir, ficando assim a informação contida no grafo correcta. Num caminho inverso, quando o HUA recebe um pedido para associar um novo TEA vizinho, este verifica qual o HUA que consegue alcançar apartir desta nova passadeira, e passa a incluí-la como novo ramo do grafo, partilhando esta informação com os restantes HUA's.

Com esta versatilidade e, com o conjunto de possibilidades descritas no ponto anterior, o ST torna-se mais flexível e permite a sua reconfiguração a qualquer momento.

3.5. *Transport Entity Agent*

Como já foi referido, o ST tem duas entidades principais, com o intuito de controlar encaminhadores e passadeiras que o constituem, estes últimos são da responsabilidade da entidade, em que as suas capacidades e funcionalidades passam a ser descritas neste subcapítulo.

As passadeiras são entidades mecânicas que possibilitam o deslocamento do produto ao longo de uma linha (ou um ramo do grafo que a descreve). Cada uma das passadeiras tem, obrigatoriamente associados como vizinhos, duas entidades mecânicas do tipo encaminhador, abstraídas cada uma delas por um HUA, esta obrigatoriedade deve-se ao facto de uma passadeira sem ponto de entrada ou sem ponto de saída ser totalmente inútil ao sistema.

Para além destas características, a passadeira tem a habilidade de permitir o acoplamento das estações, onde os RA's e CLA's vão disponibilizar a prestação de habilidades, em pontos onde isso seja fisicamente possível.

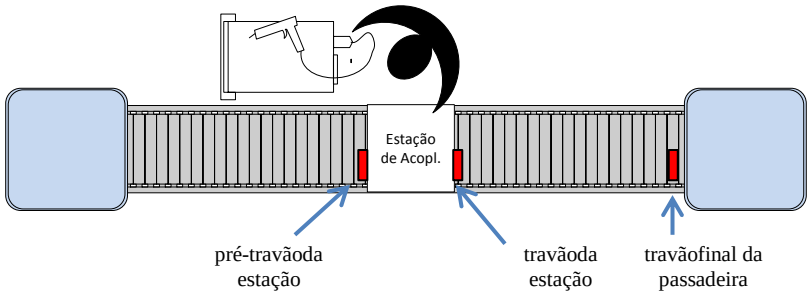


Figura 10 -Exemplo de uma passadeira com uma estação acoplada

No exemplo da Figura 10 está representada uma passadeira, constituída apenas por uma estação, operada por um humano. Esta estação está acoplada na única estação de acoplamento disponível neste exemplo. Para permitir um controlo do fluxo dos produtos na passadeira, esta tem de ter um travão no final, permitindo controlar a passagem de paletes para o encaminhador, sendo este travão atuado quando for dada ordem para que a palete seja enviada, e dois travões por cada estação, um para bloquear o produto na estação, durante a execução das habilidades e um para controlar a entrada na estação, das paletes com o produto, resultando num total de $\{2 \times s + 1\}$ travões necessários, em que s é o número de estações acopladas.

3.5.1. Associar uma Passadeira ao Sistema

Quando se pretende associar uma nova passadeira ao ST, é necessário que os HUA's que serão seus vizinhos, já estejam em execução. Verificada esta obrigatoriedade o TEA, que

tem a responsabilidade de controlar a passadeira, comunica aos HUA's vizinhos a associação de um novo vizinho esta comunicação deverá ter a informação, se esse HUA é de entrada ou de saída da passadeira.

Neste ponto da execução a passadeira é adicionada, e comunica-o sem qualquer estação acoplada, logo sem habilidades disponíveis, só após esta inicialização e associação aos vizinhos é que o acoplamento das estações pode ser realizado, e assim comunicada a existência de novas habilidades disponíveis.

Para além do TEA comunicar aos HUA's, que é vizinho destes, e se estes são de entrada ou saída, é necessário comunicar ao HUA de entrada que este tem a possibilidade de alcançar um novo HUA, que neste caso é o HUA de saída.

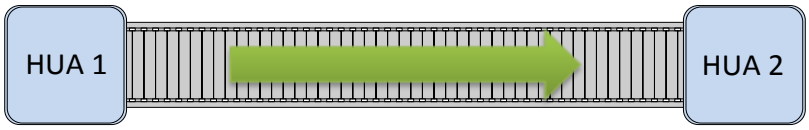


Figura 11 – Exemplo de passadeira vizinhos

Neste caso (Figura 11), aquando da associação desta nova passadeira, o TEA comunica ao HUA 2 que este tem um novo vizinho, e que o seu encaminhador é o ponto de saída da nova passadeira. Por outro lado o TEA comunica ao HUA 1 que este é o ponto de entrada de produtos deste novo vizinho e que este HUA consegue encaminhar produtos para o HUA 2, tornando assim este nó do grafo alcançável, em que a ligação é feita por esta nova entidade.

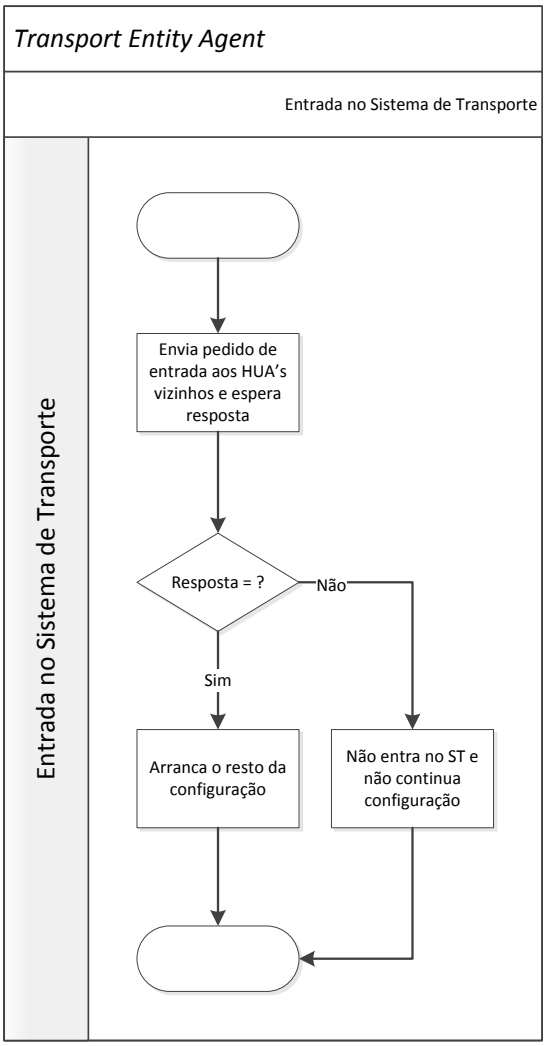


Figura 12 – Associação de um TEA ao ST

Só é permitido ao TEA arrancar e associar-se ao sistema caso os HUA's que serão seus vizinhos tenham sido configurados e inicializados. Quando o TEA é arrancado, este comunica de imediato a sua presença aos HUA's (Figura 12), comunicando-lhes quais as suas funções, se ponto de entrada ou saída deste TEA.

3.5.2. Acoplamento e Desacoplamentode Estações

O acoplamentode estações permite que, em qualquer altura, o utilizador possa associar estações a um ponto da passadeira, tornando assim acessível uma nova estação no sistema. Estação essa que terá RA's e/ou CLA's associados, que poderão executar as habilidades para os quais estão habilitados.

O TEA está habilitado a que se possa efetuar acoplamento e desacoplamento de estações, em locais onde isso seja praticável, de fato o acoplamento das estações só poderá ser efetuado, como já vimos no ponto anterior, numa zona da passadeira onde estejam livres um

travãoe um pré-travão. A estas zonas, onde é possível fazer os acoplamentos das estações, chamou-se de pontos de acoplamento.

Duas entidades da arquitetura geral do sistema estão habilitadas a registar habilidades numa estação de acoplamento, sendo estas o agente RA e o agente CLA. Quando uma estação é inicializada e acoplada a um ponto deste género, todos os RA's e CLA's contidos naquela estação devem comunicar a sua disponibilidade, naquele ponto do sistema, ao agente do tipo TEA, ou seja, é como se cada RA e CLA, fizessem o seu próprio acoplamento naquela posição da passadeira.

Como se pode ver na Figura 13, existem algumas etapas, até que o acoplamento de um RA ou CLA constituinte de uma estação, esteja completo, visto que é preciso ter em conta o grande número de entidades constituintes do ST que é necessário notificar sobre esta alteração ao nível de um TEA.

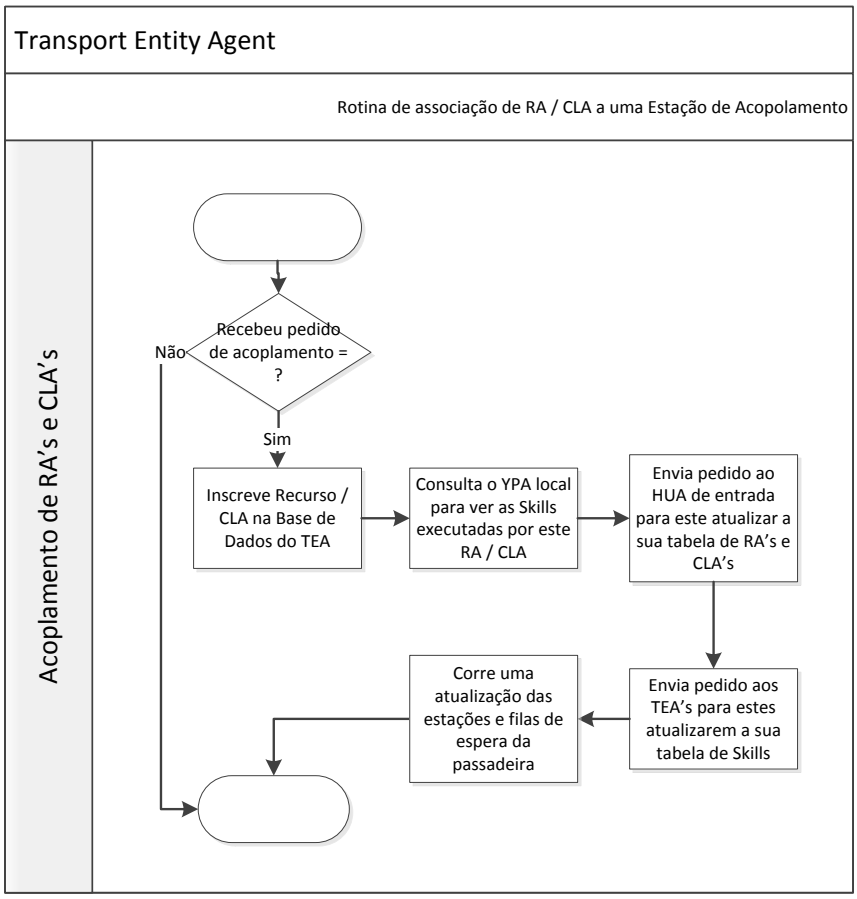


Figura 13 – Comportamento do TEA aquando de um pedido de acoplamento

Aquando de um pedido de acoplamentoa um TEA, este imediatamente inscreve o RA ou CLA na sua base de dados (BD) de entidades acopladas a uma das suas estações de acoplamento, mas neste momento sem qualquer tipo de habilidades associadas a esta entidade,

pois essa consulta ainda não foi efetuada nem essa informação é comunicada aquando do envio do pedido. Estando essa inserção concluída, o TEA irá consultar o YPA local, do TEA onde está a ser feito o acoplamento, onde o RA ou CLA que está a pedir para se ligar naquela estação de acoplamento já inscreveu as habilidades que pode executar. Com esta consulta o TEA fica a saber as habilidades que aquele RA ou CLA pode executar e associa essas habilidades à BD, onde previamente tinha associado a entidade, mapeando assim as habilidades à entidade. Agrupada que está a informação, o TEA consegue saber para cada uma das estações que estão presentes, quais os RA's e/ou CLA's lá alocados e que habilidades poderão ser executadas.

Após a informação, relativa ao novo agente a ser alocado, ter sido tratada localmente, ou seja, ao nível do TEA responsável pela passadeira onde esta entidade irá operar as suas habilidades, é necessário informar todo o resto do ST sobre esta alteração. Numa primeira fase o TEA que recebe o pedido de acoplamento comunica ao HUA vizinho responsável pela entrada de produtos, qual o RA ou CLA que se está a alocar a aquele TEA.

O mesmo tipo de comunicação é iniciado com todos os outros TEA's, com a mesma finalidade, de tornar conhecidas as alterações realizadas na passadeira onde o acoplamento é concretizado, este conhecimento vai ser necessário para tomadas de decisão relativas ao encaminhamento do produto.

Quando um TEA arranca não tem qualquer tipo de estação, isso só acontece posteriormente, por via dos pedidos efetuados pelos RA's e CLA's pertencentes. Embora as estações de acoplamento existam, ainda não foi requerido acoplamento de qualquer estação, fazendo com que a passadeira tenha apenas um travão final e seja abstraída por uma única fila de espera. Após a alocação de um primeiro RA, a uma estação de acoplamento, é necessário reajustar a estrutura interna do TEA, fazendo um novo mapeamento das filas de espera, consoante as posição das estações e as posições relativas dos pré-travões, relativamente às estações.

Sendo assim as diferenças encontradas numa passadeira,antes e depois de se inicializar uma estação, estão representadas na Figura 14.

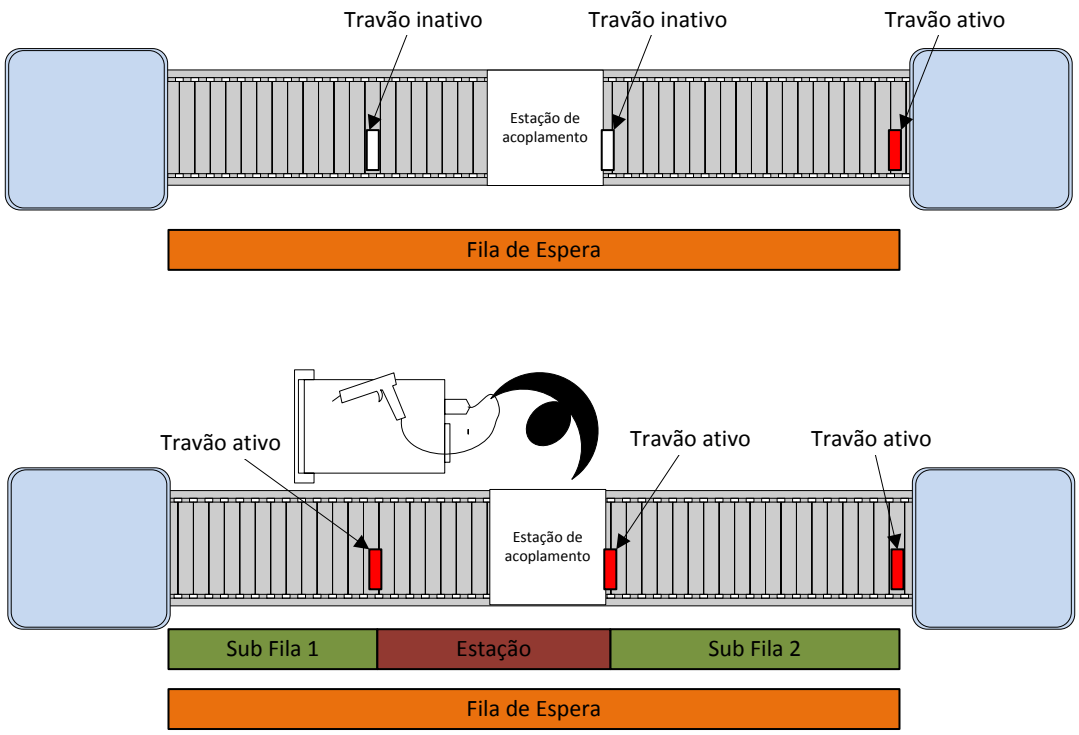


Figura 14 - Alterações verificadas na passadeira após a inicialização da estação

Para colmatar as dificuldades verificadas, no controlo do fluxo de produtos na passadeira, devido às constantes alterações na sua constituição, foi proposto uma modelação de dados que consiste na criação de sub filas, para além da fila de esperaprincipal, para que o TEA saiba, em cada secção da passadeira, que PA's estão e qual a sua ordem.

Com a abordagem das sub filas, o TEA ao alocar uma estação ainda por inicializar, este automaticamente executa uma renovação à sua estrutura e verifica onde se encontram as estações. Durante a renovação da estrutura interna, é construído um modelo de dados, constituído por uma fila de espera principal e sub filas, que representam no nível lógico as filas de espera, atrás de todos os pré-travõese atrás do travão final da passadeira. Também as estações fazem parte desta estrutura de dados, mas sem ser necessário recorrer a filas de espera, pelo fatodas estações apenas poderem ter um produto de cada vez, sendo apenas preciso verificar se tem produto ou não, e qual o mesmo. O número de sub filasé facilmente calculado, sendo retirado da expressão $\{s + 1\}$, em que s é o número de estações presentes na passadeira.

3.5.2.1. Estrutura Interna

De forma a permitir a gestão da área da passadeira, é necessário ter em conta algumas medidas de segurança, tornando segura a gestão das filas de espera, sem que a proximidade excessiva entre estações e, a distânciaentre estações e encaminhadores possam interferir ou até impossibilitar o correto fluxo de produto, como se pode ver na Figura 15.

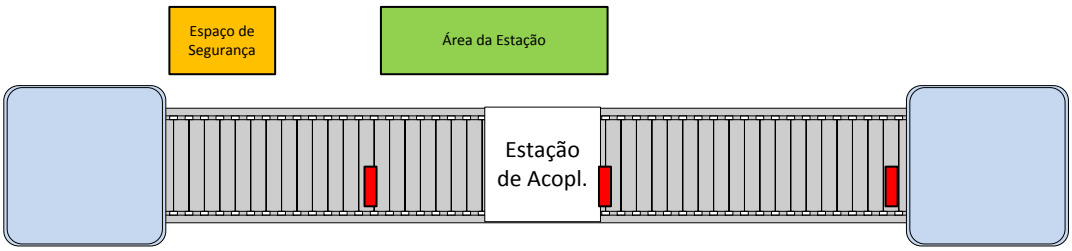


Figura 15 – Distâncias de segurança a tomar em conta

Com estas medidas de segurança é possível saber qual o tamanho seguro para cada uma das sub filas, não havendo a possibilidade de dar entrada a um produto numa das sub filas e esta não ter espaço físico para o alocar, o que se traduzia numa falha grave.

3.5.2.2. Controlo do Fluxo de Produtos na Passadeira

Durante a execução de uma entidade do tipo TEA é necessário, frequentemente, renovar a informação sobre a ocupação da passadeira, relativamente a quais os produtos presentes na fila de espera deste, e em cada uma das sub filas de espera e estações.

Durante o fluxo de produtos na passadeira,o TEA tem a responsabilidade de saber qual o destino de cada um deles, pois este poderá ser uma das suas estações ou não, caso não o seja, o produto apenas seguirá o caminho normal pela passadeira, sem uma paragem obrigatória, até ao final da passadeira, para que o produto possa ser enviado para o encaminhador de saída e aí ser novamente encaminhado para outra passadeira.

No caso de o produto ter como objetivo operar uma habilidade numa das estações da passadeira, aquando da chegada desse mesmo produto à estação de destino, o TEA terá de bloquear o produto durante as negociações com o ST, RA ou CLA, e também durante a execução da habilidade.

Em condições normais, esta será a única ocasião onde o ST pretende “atrasar” o despacho de um produto, pois em todos os casos o objetivo prioritário do ST, no que respeita ao fluxo das paletes,é encaminhá-las o mais rapidamente possível.

No seguinte exemplo, Figura 16, em que os produtos estão representados a amarelo, a passadeira está completamente preenchida, não podendo receber mais nenhum produto, embora exista um espaço entre o produto da estação e o seu pré-travão. No espaço reservado à estação apenas poderá existir, em cada momento, um único produto, salvaguardando a não ocorrência de qualquer perturbação aquando da execução de uma habilidade.

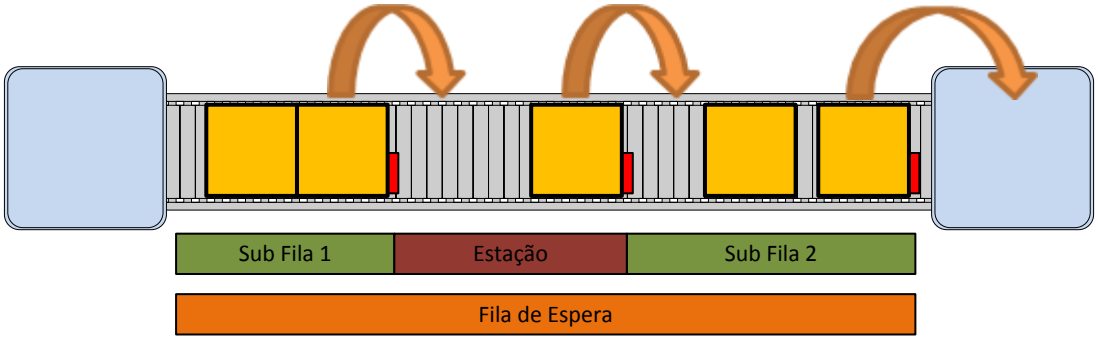


Figura 16 - Exemplo de ocupação de uma passadeira

De uma forma rotineira, o TEA verifica a presença de produtos parados nos travões da passadeira, quer sejam travões, pré-travões ou travão final. No caso dos travões que limitam logicamente o final das subfilas de espera, sendo estes os pré-travões das estações e o travão final da passadeira, sempre que um produto se encontra num desses travões, o objetivo é que este imediatamente passe para a próxima secção, seja ela uma estação ou um encaminhador, este envio está dependente em ambos os casos de presença ou não de algum produto no ponto mais à frente, ou seja, presença de algum produto na estação ou encaminhador de destino.

No caso da estação, o TEA poderá não encaminhar o produto para a próxima subfila de espera, devido ao fato de o produto estar num destes casos:

- A ser executada uma habilidade;
- Durante uma negociação do PA com o ST;
- Durante uma negociação do PA com RA's e/ou CLA's;

Todas estas fases serão descritas no próximo ponto.

3.5.3. Negociações com o Produto

A maioria das interações dos agentes que constituem o ST, excluindo as interações entre agentes do mesmo tipo, serão com o PA, quer seja para facultar informação requerida pelo produto como responder ao pedido de encaminhamento por parte do mesmo.

Na Figura 17 estão descritas as interações realizadas entre o PA e o TEA responsável pela passadeira onde se encontra o produto.

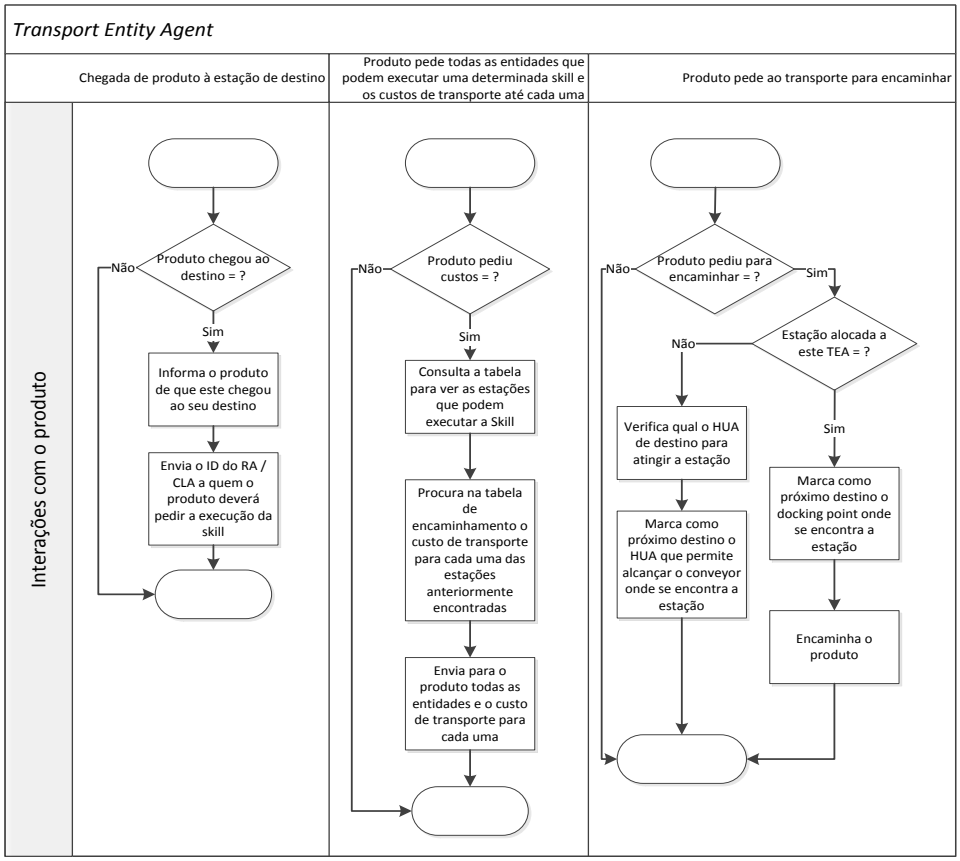


Figura 17 – Esquema representativo das interacções entre TEA e PA

A interação descrita mais à esquerda é de simples descrição pois a única função é informar o PA, de que chegou ao destino e enviar-lhe também o identificador do RA ou CLA com quem este deverá comunicar, para que lhe seja executada a habilidade pretendida.

Posteriormente, quando o PA terminar a execução de uma habilidade, este encontra-se numa estação pertencente a um TEA. O PA pergunta ao TEA quais os RA's e/ou CLA's disponíveis para executar a próxima habilidade do seu plano. O TEA, através da consulta à sua BD, envia para o PA uma lista com todas estas entidades e custo de alcançar cada um deles, baseando-se no custo do ST. O PA com esta informação negoceia com todos eles, terminada a negociação, o PA decide o próximo destino e pede ao TEA onde se encontra para este iniciar o encaminhamento do mesmo. Este comportamento está descrito no fluxograma central.

O fluxograma à direita representa o comportamento do TEA aquando de um pedido de encaminhamento do PA. Quando o PA pede o início do encaminhamento, o TEA verifica se a estação de destino está acoplada nele próprio, ou seja, que é pertencente à passadeira que este abstrai. Caso este situação se verifique, o TEA marca como destino o número que identifica a estação de acoplamento, para que este possa verificar a sua chegada ao destino.

Caso a estação requerida pelo PA não se encontre no TEA atual, este consulta a sua TE e a BD (com os HUA's e as estações alcançáveis por estes) e verifica qual o próximo HUA para o qual deverá encaminhar.

3.5.4. Tabela de Encaminhamento

Como referido no ponto anterior e à semelhança do HUA, o TEA necessita de consultar uma TE. Esta tabela é utilizada duas vezes, aquando do pedido dos custos de encaminhamento até às estações e quando o PA pede para se iniciar o encaminhamento até uma estação.

De forma a que o TEA tenha uma TE sempre atualizada, este utiliza uma cópia integral da TE do HUA de saída. A TE do HUA pode ser utilizada pois o TEA tem obrigatoriamente de encaminhar o PA até esse HUA. Como o primeiro destino (e onde será tomada a primeira decisão relativamente ao encaminhamento) é sempre o HUA de saída, a TE deste coincide com a TE do TEA.

Com esta informação o TEA tem toda a informação necessária à negociação com o PA, tanto no pedido de custos de todos as estações que podem executar uma habilidade, como para a inicialização de um novo encaminhamento.

3.5.5. Métrica e o seu Impacto

Com o objetivo de alcançar um destino de forma rápida e eficaz, o ST tem associada uma métrica que ajusta o custo de atravessar cada ramo do grafo. Esta métrica tem um impacto significativo no desempenho, pois reflete-se num balanceamento da carga presente em cada passadeira. Este balanceamento da carga tem como objetivo evitar o congestionamento das passadeiras.

O TEA rotineiramente calcula o custo de ser atravessado e envia-o ao HUA de entrada de produtos na passadeira. O custo recebido pelo HUA é utilizado para renovar a informação do TEA vizinho. Com esta informação o HUA de entrada sabe que tem um novo custo para alcançar o HUA que este TEA alcança. De um ponto de vista lógico resulta num novo custo de encaminhamento entre estes dois nós do grafo.

Numa próxima comunicação com os restantes HUA's a topologia do sistema já será atualizada e assim todos os HUA's calculam as suas TE's baseada nesta atualização.

O TEA disponibiliza as seguintes variáveis para o cálculo da métrica.

- Velocidade de transporte da passadeira (VP);
- Comprimento da passadeira (CP) em número de unidades;

- Número de estações acopladas (NE);
- Número de produtos no preciso momento (NP);

Com este conjunto de dados é possível atualizar o custo associado a cada ramo do grafo, fazendo com que o encaminhamento seja rápido, eficaz.

A expressão $Custo = a \times \mathrm{CP} + b \times NE + c \times NP$, é utilizada para o cálculo do custo associado a cada passadeira, em que *a*,*b* e *c,* são constantes utilizadas para ajuste da preponderância e impacto de cada parcela. Como é perceptível, através de uma breve análise à expressão anterior, verifica-se que a CP penaliza as passadeiras com maior comprimento e menor desempenho no encaminhamento, enquanto NP penaliza as passadeiras com mais produtos, com o objetivo de balancear a carga entre passadeiras.

Com esta expressão é possível controlar a preponderância dos três fatores usados. No caso de o objetivo ser um balanceamento máximo da carga, o fator *c* é usado para o efeito, tendo um valor muito superior aos outros. Tendo como referência também a importância do NE, devido aos atrasados verificados neste ponto da passadeira, pela atuação dos travões, o fator *b* poderá ter um grande impacto no desempenho do sistema. O CP é importante devido a uma possível existência de passadeiras muito maiores que as restantes no ST, e assim serem penalizadas no cálculo do custo.

3.5.6. Capacidade de Adicionar e Remover Paletes do Sistema de Transporte

A arquitetura suporta que em qualquer altura, durante a execução, uma palete possa ser removida ou adicionada ao sistema numa posição diferente. O sistema deverá ter a capacidade de se auto organizar de forma a conseguir reagrupar os produtos nas diferentes sub filas,de forma correta, correspondendo ao sistema real.

Para que esta característica seja implementada, o ST deverá estar munido de leitores de *Radio-Frequency Identification* (RFID) em todas as estações do sistema, como no final de cada uma das passadeiraspresentes, esta imposição não deverá ter exceções, pois inviabiliza a implementação desta capacidade.

A remoção e adição de paletes só é permitida, pela arquitetura, nas sub filas presentes na passadeira, ou seja, num destes três espaços:

- Desde o início da passadeira até ao pré-travão da primeira estação;
- Desde um travão de uma estação até ao pré-travão da estação que se segue;
- Desde o travão da última estação até ao travão final da passadeira;

Na Figura 18 está representado um exemplo de uma linha. No exemplo estão representadas a vermelho as áreas onde estas operações não são aceites e a verde onde estas poderão acontecer.

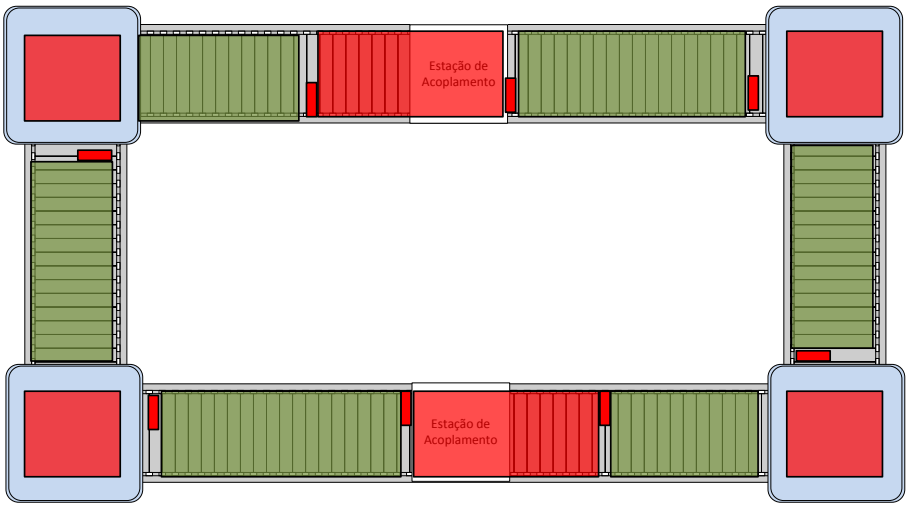


Figura 18 - Esquema representativo das zonas em que são permitidas mudanças manuais na posição da palete

Como os leitores de RFID encontram-se apenas nas estações, a verificação é feita na estação. A sub fila está frequentemente a encaminhar paletes para a estação que lhe precede. Quando a palete chega à estação, esta verifica se o PA alocado a aquela palete é o mesmo que a sub fila pediu para entrar na estação. Caso o PA não seja o esperado, a estação verifica se este está ainda presente na sub fila anterior ou se é um novo. Caso este se encontre na sub fila anterior é porque se deu a entrada de uma nova palete, adicionada numa posição exatamente à frente do PA que agora era esperado. Caso contrário foi removida a palete onde estava alocado o PA que agora era esperado. Em ambas as situações a sub fila anterior é reorganizada de forma a esta representar o sistema real novamente, com a correta sequência de PA's.

O outro caso onde esta situação pode ocorrer é no final da passadeira, onde à semelhança das estações se encontra um leitor de RFID. Quando uma palete chega ao final da passadeira, antes de se iniciar a comunicação para o encaminhamento desta para o HUA de saída, o TEA verifica se o PA é o esperado. Caso o PA não seja o esperado esta última sub fila que constitui o TEA é reorganizada e o novo PA adicionado à última posição da sub fila.

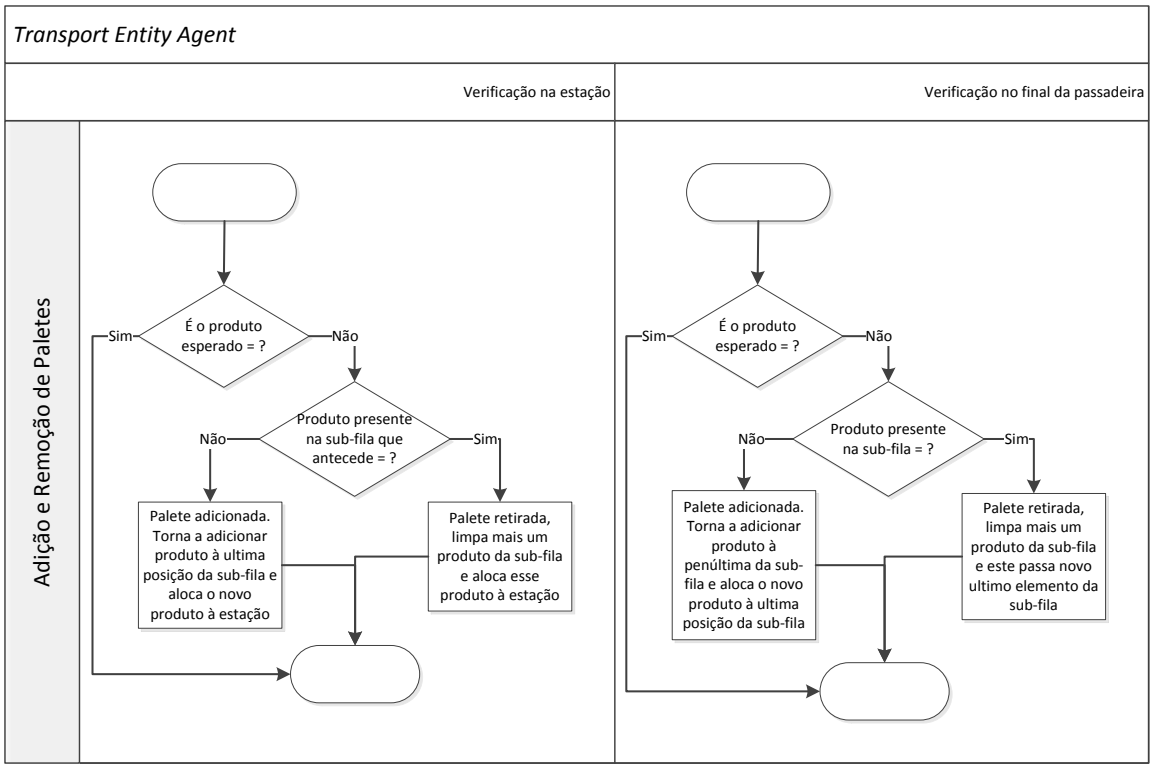


Figura 19 - Comportamento do TEA aquando da adição ou remoção de paletes

Na Figura 19 estão representados os comportamentos já anteriormente referidos. Estes comportamentos permitem assim ao ST reagir a adições e remoções de paletes durante a execução, reorganizando as sub filas de espera.

4. Implementação

Para a implementação da arquitetura proposta recorreu-se à linguagem de programação Java em conjunto com uma plataforma orientada ao comportamento, JADE (Java Agent Development Framework) (Bellifemine, Poggi et al. 1999).

4.1. *HandoverUnitAgent*

4.1.1. Cálculo da Tabela de Encaminhamento

Tendo em vista o cálculo da TE, guardada e calculada em todos os HUA's, é necessário primeiramente apreender qual a topologia do ST, e então posteriormente calcular a sua TE, tendo como base essa topologia.

As interações entre cada um dos intervenientes neste processo, para que se consiga alcançar estes objetivos, estão descritas no diagrama de sequência da Figura 20.

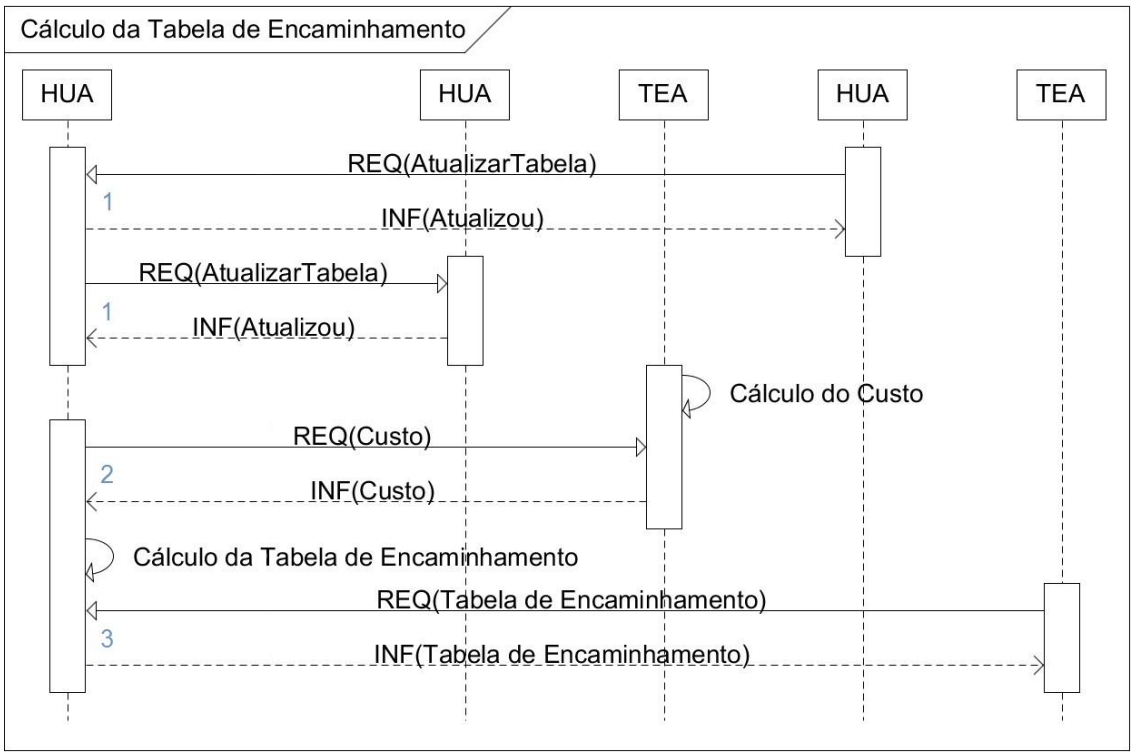


Figura 20 – Diagrama de sequência dos agentes de transporte durante o cálculo da tabela de encaminhamento

O protocolo utilizado para a troca das mensagens entre agentes é o *FIPARequest*. Na imagem o tipo de mensagens é representado com as letras iniciais. No caso de REQ indica ser um *Request* e no caso de INF indica ser um *Inform.*

No ponto assinalado com 1 os HUA's iniciam um pedido de atualização da TE, esta situação acontece periodicamente. O HUA quando envia pedido informa também os outros HUA's da topologia da rede, neste caso dos HUA's alcançáveis por este e qual o custo associado.

No ponto 2 o HUA pede ao TEA, vizinho deste, o seu custo. Assim o HUA atualiza o custo de alcançar o HUA alcançável através deste TEA. No ponto 3 o HUA processa o pedido de envio da TE por parte do TEA, assim o HUA serializa a sua TE e envia para o TEA vizinho.

4.1.1.1. Conhecer a Topologia do Sistema de Transporte

De forma a guardar a tipologia da rede, trocada entre os vários HUA's do ST, baseada na informação dos TEA's vizinhos de cada um deles, recorreu-se a uma BD.

Essa BD foi implementada com o recurso a uma *HashMap*. Na *HashMap* a chave de cada entrada é o *localname* do HUA, cada agente tem um *localname* único na plataforma, funcionando como um identificador. A informação recolhida sobre cada HUA é guardada na *HashMap* sobre a forma de um objeto *DiverterInfo*. Este objeto é constituído por:

- O AID (identificador único para cada agente) do HUA;
- A data da última atualização;
- Uma lista com os vizinhos diretos deste, ou seja, quais os HUA's que este consegue alcançar e a que custo.

Com esta BD, o HUA tem em sua posse, toda a informação necessária à criação da topologia do sistema.

Cada HUA como foi visto anteriormente, necessita ter a informação referente aos TEA's vizinhos. Nesta informação deverá constar o custo de o utilizar e o HUA que pode ser alcançado recorrendo a este vizinho.

Para guardar esta informação implementou-se uma *HashMap* em que a chave das entradas é o *localname* do vizinho. Nessa *HashMap*são guardados objetos do tipo *ResourceConveyor*, constituídos por:

- AID do vizinho;
- Uma variável que indica se este TEA é de saída ou entrada;

- Um objeto do tipo *OutputDiverter*,com o HUA alcançável. Neste objeto temos toda a informação relevante do HUA de destino como o AID e o custo de o alcançar;

De forma a tornar possível esta troca de estados entre HUA's recorreu-se à implementação dos *behaviours*, representados na Figura 21.

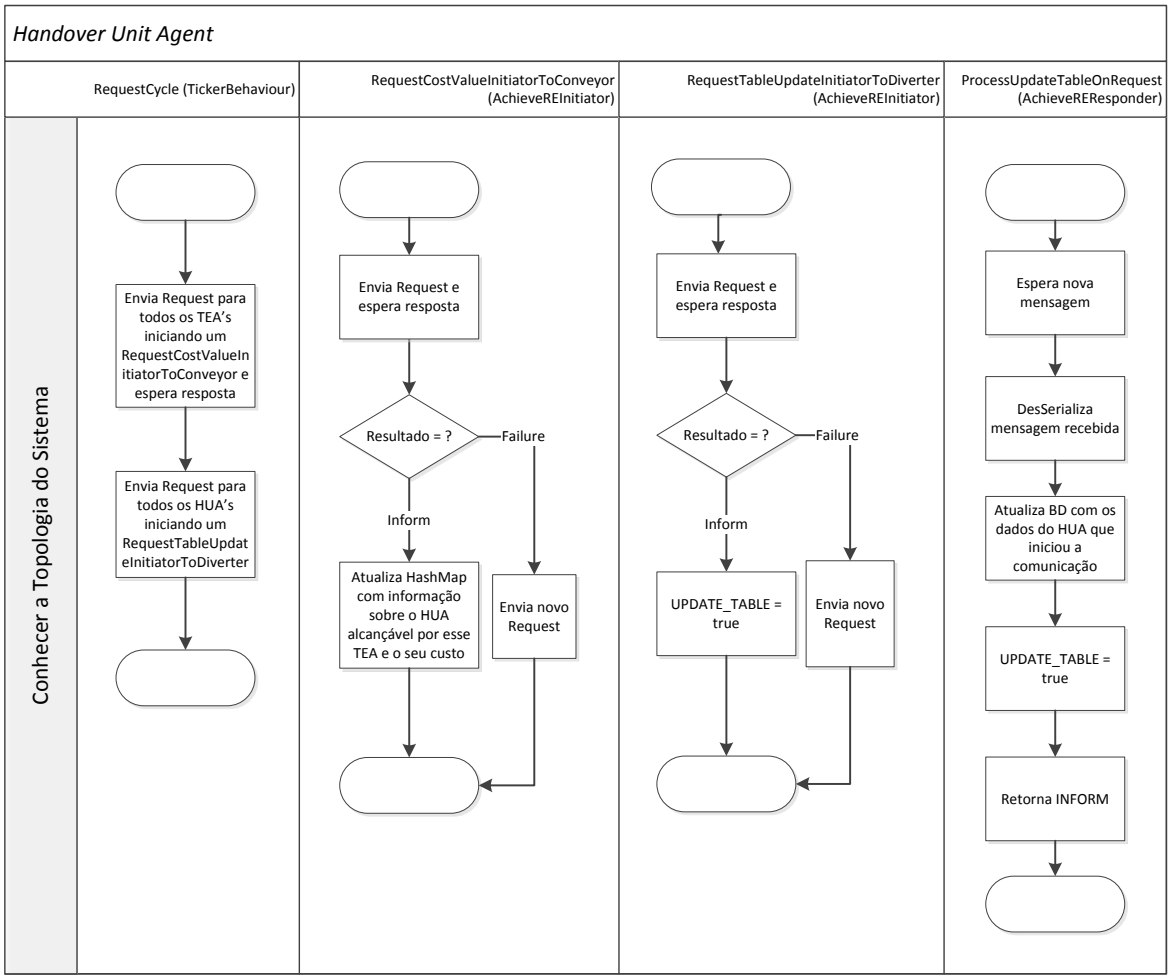


Figura 21–*Behaviours* responsáveis pelo reconhecimento da topologia do sistema

Neste caso a implementação de um comportamento repetitivo, é permitido recorrendo-se ao *TickerBehaviour*. Este *behaviour* é executado de uma forma repetitiva, com um intervalo entre execuções definido no lançamento do *behaviour*. No caso do *RequestCycle behaviour*, este é injetado logo na configuração inicial do agente e com um tempo de intervalo entre "*tickers*" de mil milissegundos.

Quando o *behaviour*, descrito anteriormente é despoletado, inicializa-se uma comunicação, primeiro com todos os TEA's (ponto 2 da Figura 20), e posteriormente com todos os HUA's (ponto 1). Para tornar possível esta comunicação recorreu-se à implementação de *behaviours* do tipo *AchieveREInitiator*. Este tipo de implementação permite inicializar

comunicações com outros agentes, garantindo o cumprimento do protocolo *FIPARequest*, responsável pela correta conversação entre agentes.

No caso da troca de informação com os TEA's, este ao receber uma resposta afirmativa, ou seja, o *Inform* a confirmar que o TEA executou o seu pedido, é atualizada a informação relativa a aquele TEA, na *HashMap*, atualizando o *ResourceConveyor*. Caso a resposta seja um *Failure*, resultado da não execução do pedido, o HUA que tinha inicializado a comunicação torna a enviar o mesmo pedido.

No *behaviour* inicializado com os restantes HUA's, o *RequestTableUpdateInitiator*, o HUA não só envia a sua informação relativa à topologia, como também pede aos HUA's restantes que recalculem a TE. Quando a resposta é afirmativa, o HUA que enviou a atualização força o valor *true* variável *UPDATE_TABLE*, responsável por assinalar a necessidade de atualização da TE. Caso contrário, um novo pedido é enviado, à semelhança do que acontece na comunicação com os TEA's.

Do lado de quem recebe, os HUA's ficam em espera até que as mensagens sejam processadas pelo *behaviour* do tipo *AchieveREResponder*. Este *behaviour* é arrancado na configuração inicial do agente. Quando a mensagem é processada, atualiza-se a *HashMap* com a topologia da rede e força também a *true* a variável de controlo *UPDATE_TABLE*, de modo a que assim que possível a TE seja atualizada.

4.1.1.2. Cálculo da Tabela de Encaminhamento

Conhecida a topologia, recorrendo aos comportamentos descritos no ponto anterior, é agora objetivo o cálculo do melhor caminho, sendo esse o caminho com menor custo associado.

Em cada ponto da execução do sistema, cada HUA tem uma TE, em que baseia as suas decisões. Essa TE é fundamentada nos caminhos mais curtos. De forma a possibilitar o cálculo da TE, e como já foi explicitado no capítulo da arquitetura, é necessária a implementação do comportamento responsável pela execução do AD.

Existem duas caraterísticas muito importantes a ter em atenção durante a implementação do *behaviour* que suporta o cálculo da TE. O cálculo da TE acontecerá com muita frequência, devido ao elevado número de pedidos enviados pelos outros HUA's para que este se atualize e esta atualização não poderá bloquear o restante processamento do HUA, como o encaminhamento de paletes.

De modo a que este *behaviour* não bloqueie o agente, recorreu-se a um *TickerBehaviour* com um intervalo entre *tickers* muito baixo. Assim, quando este *behaviour* é o próximo do escalonamento, este corre uma iteração do AD, caso este esteja em curso. Caso contrário não há nada a fazer. Para controlar se a execução do AD está a decorrer consulta-se a variável *RUNNING_DIJKSTRA*.

O *TickerBehaviour* responsável por este cálculo está representado na Figura 22.

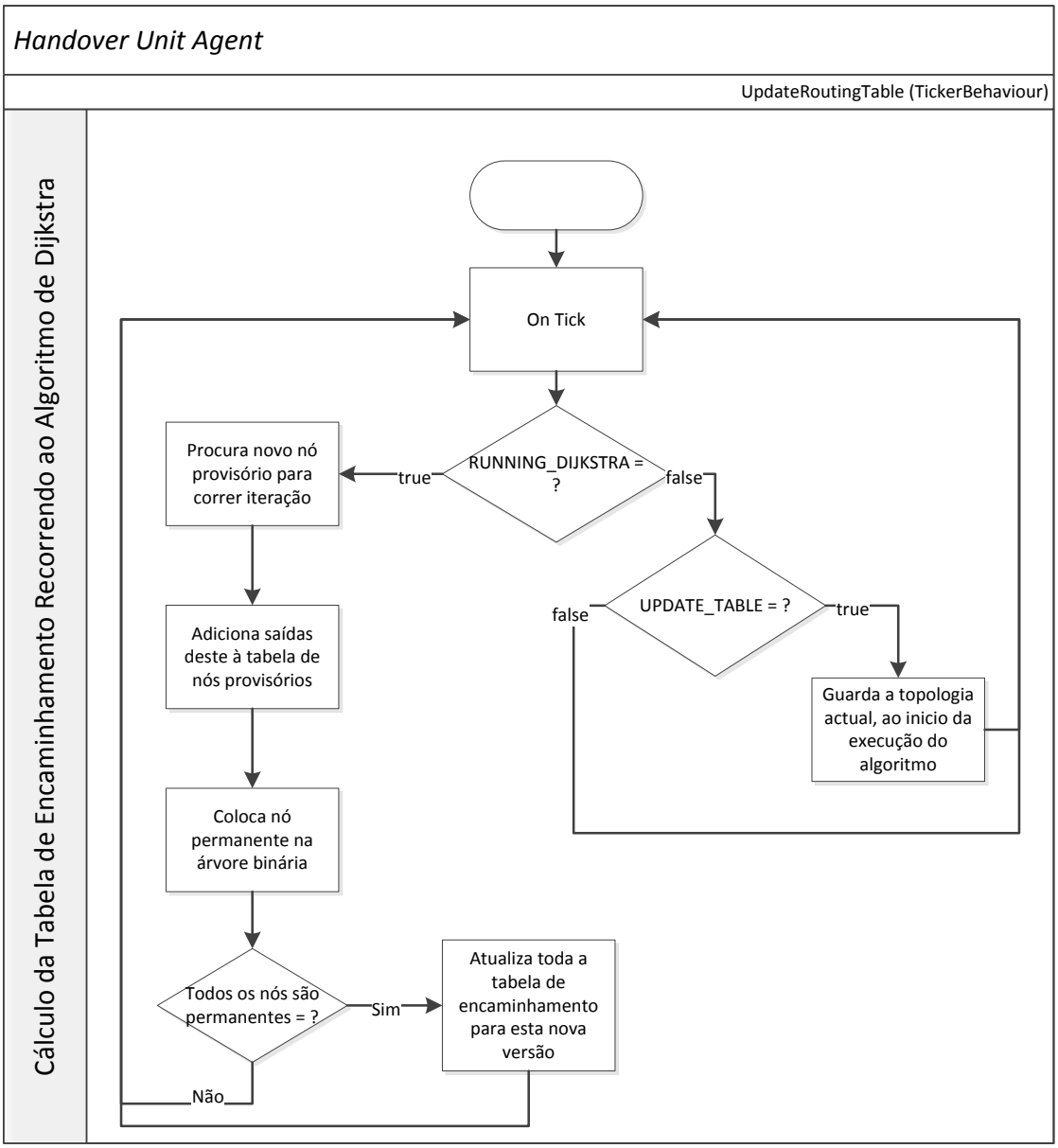


Figura 22 – *Behaviour* responsável pelo cálculo da TE

Este *behaviour* é inicializado assim que o agente inicia a sua execução, com um intervalo entre "*tickers*" de 500 milissegundos. Este valor foi ajustado de forma a ser curto o suficiente para que a TE seja atualizada de forma rápida, mas também de forma a não prejudicar o desempenho da execução do agente.

Quando este *behaviour* é disparado, verifica-se se está a meio da execução do AD, pois se se estiver num estado intermédio da execução do algoritmo este irá correr mais uma iteração do algoritmo, procurando mais um nó provisório e verificando todas as suas saídas, no final atribui-se um novo nó permanente à tabela de nós permanentes. Quando todos os nós, que

compõem o grafo, estiverem dados como permanentes, a execução do algoritmo está terminada e como tal a TE está atualizada.

Durante a execução do algoritmo, os dados originais são preservados para que possam ser atualizados. A sua atualização poderia provocar incongruências que resultariam num cálculo da TE errado, e como tal é utilizada uma cópia da BD, relativa à topologia, efetuada no início do cálculo.

De forma a tornar possível o encaminhamento por parte do HUA, durante o cálculo da TE, a tabela deste só é atualizada aquando do término do seu cálculo, deixando a versão anterior como base das tomadas de decisão, nesse estado intermédio.

4.1.2. Encaminhamento do Produto

O encaminhamento do produto, ao nível do HUA é feito em duas etapas, quando se dá entrada deste, proveniente de uma passadeira a pedido do TEA que a abstrai, e posteriormente quando é encaminhado para uma nova passadeira, em pedido é realizado pelo HUA ao TEA vizinho.

Cada encaminhador, tem apenas a capacidade de armazenar simultaneamente um produto, como tal, o HUA só pode aceitar a entrada de um novo aquando da saída do existente, resultando numa situação simples de ocupado ou não ocupado, sem se dar acomodação de vários produtos.

O comportamento pretendido, com a implementação desenvolvida, tem o seguinte aspecto, explicitado na Figura 23.

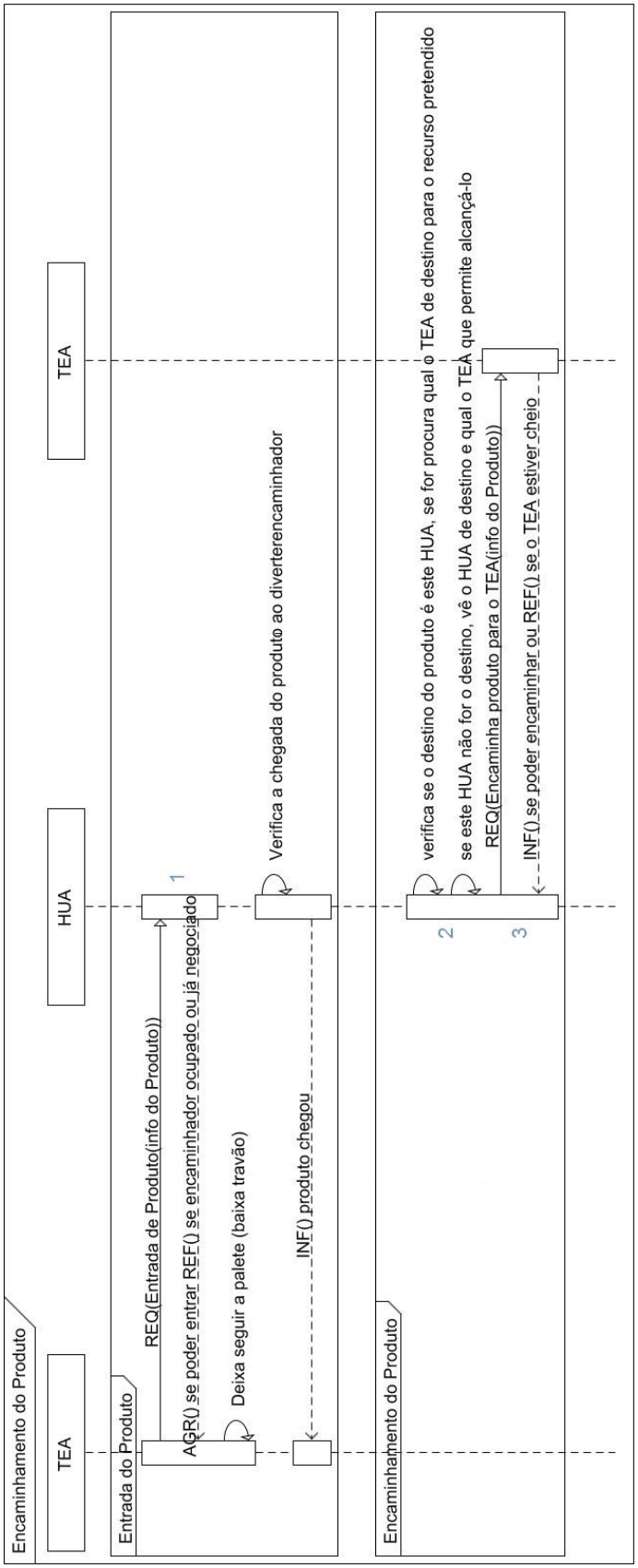


Figura 23 – Sequência de execução de entrada e saída de produtos no HUA

Na entrada do produto no encaminhador (ponto 1 da Figura 23) o HUA recebe o pedido do TEA. No caso do HUA estar disponível para receber a palete envia uma primeira mensagem de aceitação e posteriormente uma mensagem confirmando a chegada da mesma.

Para que o HUA encaminhe o produto para um novo TEA, seu vizinho, este verifica qual o destino do PA (ponto 2 da Figura 23) e posteriormente inicia a negociação com o TEA, de forma a poder despachar a palete (ponto 3 da Figura 23).

4.1.2.1. Entrada de Produto no Encaminhador

Durante a sua execução, o HUA necessita estar alerta para o fato de uma passadeira de entrada propor-lhe, a entrada de um produto, como tal é necessário que este agente tenha implementado um mecanismo que permita executar esta negociação.

Quando o produto chega ao final de uma passadeira esse TEA, no estado imediatamente a seguir irá questionar o HUA responsável pelo encaminhador de saídasobre a sua disponibilidade para receber um produto (ponto 1 da Figura 23), podendo a resposta do lado do HUA ser afirmativa ou negativa, dependendo naturalmente, da possibilidade ou não disso acontecer.

A resposta afirmativa advém dos fatos de o encaminhador, encontrar-se desimpedido e de ainda nenhuma negociação ter sido aceite. Caso uma negociação deste género já tenha ocorrido e a resposta tenha sido afirmativa, haverá uma intersecção de intenções, acontecendo uma situação intolerável por parte do sistema.

Como se pode ver na Figura 24, são dois os *behaviours* implementados, que têm como função responder ao pedido do TEA. No caso do *ProcessDispatchItemFromConveyor*, para o processamento do pedido, e mais tarde enviar a confirmação, através de um *Inform*, no caso do *behaviour AckItem*.

No caso do *ProcessDispatchItemFromConveyor*, e sendo uma resposta a um pedido, utilizou-se um *AchieveREResponder* com o objetivo de receber os pedidos enviados pelo TEA, processá-los, em conformidade com a possibilidade de receber produto ou não, responder afirmativamente, através de um *Agree* e também dar início a um *behaviour* de confirmação de chegada.

O *behaviour* responsável por verificar a chegada do produto ao encaminhador (*AckItem*) é do tipo *Simple Behaviour*, em que a condição de paragem é a resposta positiva do *hardware* aquando da sua chamada, na leitura do sensor de presença da palete. Neste caso o *behaviour* é terminado e imediatamente antes de cessar a sua execução, envia um *Inform* para o TEA que anteriormente tinha enviado o pedido de entrada do produto. Para tornar possível o envio de um *Inform*, num *behaviour* que não o de recepção de mensagem. recorreu-se ao *registerPrepareResultNotification*, tornando esta situação possível.

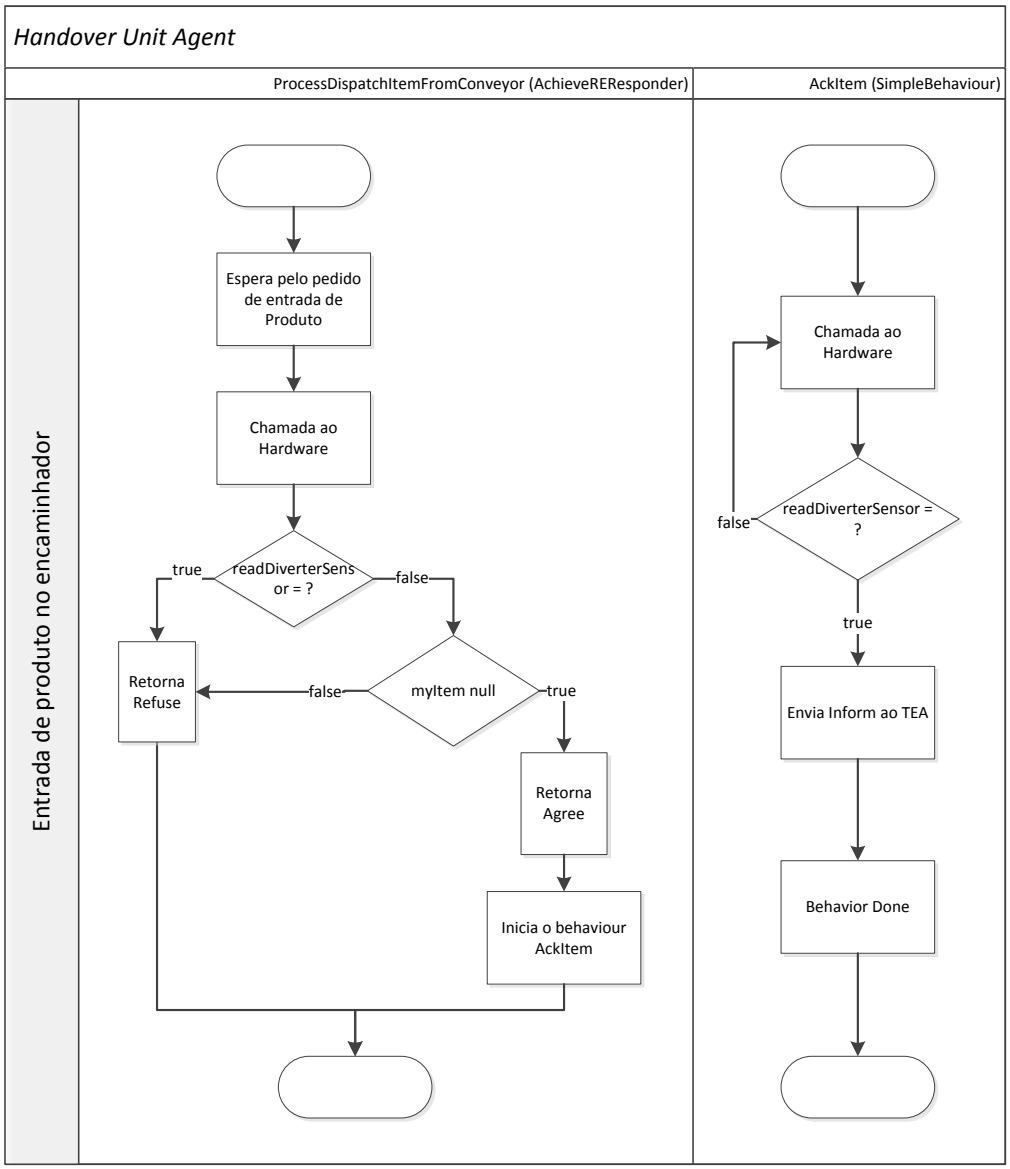


Figura 24 - *Behaviours* responsáveis pela entrada de produto no encaminhador

A verificação de existência de palete no encaminhador é feita pela chamada à função *readDiverterSensor()*, pertencente à *interfaceHardware* responsável pela chamada do mesmo. No caso da não existência de nenhuma palete negociada para a entrada o HUA não tem nenhum PA guardado na variável onde está a informação relevante sobre o mesmo. No caso de esta variável, do tipo *Item* ter o valor *null*, indica que o HUA ainda não aceitou nenhuma negociação.

4.1.3. Despacho do Produto

Posteriormente ao estado descrito no ponto anterior, ou seja, quando o produto já se encontra no encaminhador, este necessita ser encaminhado para o próximo TEA. Nesta situação podemos ter dois casos bem distintos, resultantes do fato de o produto ter como destino o HUA em que se encontra atualmente ou por outro lado o destino ser um HUA distinto.

Ambos os casos são geridos internamente pelo HUA, e como já foi referenciado, para o caso do HUA de destino não ser o presente, este consulta o HUA que será o seu *next hop* e encaminhará pelo TEA que levará até esse HUA. Na outra situação em que o HUA de destino é o atual, este terá de consultar a lista de TEA's, associados como seus vizinhos de saída e verificar qual deles tem associado o RA / CLA de destino do produto.

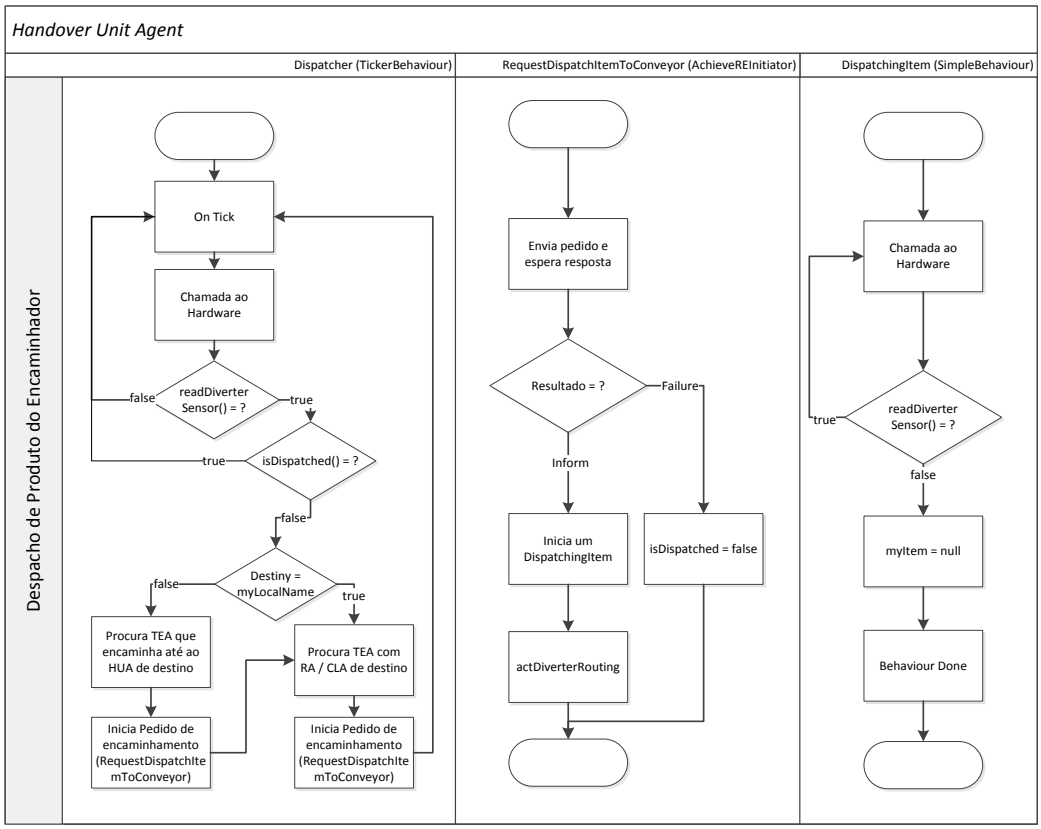


Figura 25 – *Behaviours* responsáveis pelo despacho do produto

De forma a introduzir, no *behaviour* responsável por verificar e fazer o despacho do produto, uma dinâmica repetitiva, implementou-se este através de um *TickerBehaviour*, com um período entre disparos de mil milissegundos, ou seja, a cada segundo o HUA irá verificar se tem algum produto em sua posse, caso o tenha, este irá averiguar se este já foi despachado ou não (consultando se a variável *myItem* é tem um valor diferente de *null*). Para o caso de esta ainda não ter ocorrido, o HUA irá primeiramente verificar se o HUA de destino é ele mesmo ou não, consoante essa informação este procurará o TEA de destino, quer seja para o caso de atingir o próximo HUA de destino ou para o caso da estação onde opera o RA ou CLA pretendido.

Quando se chega ao ponto de despachar a palete, onde o HUA deverá pedir ao vizinho que lhe encaminhe o produto, este dá início a um *AchieveREInitiator*, enviando o pedido. Caso o pedido seja aceite pelo TEA, o HUA atua o *hardware* tendo em vista o encaminhamento e

arranca o *behaviour DispatchingItem*, responsável por verificar o valor do sensor de presença do encaminhador. Quando o sensor indica que já não se encontra lá nenhuma palete, o HUA atualiza o seu estado para desocupado, o estado do HUA varia entre ocupado ou desocupado. A limitação de estados a estes dois deve-se ao fato de cada encaminhador apenas ter capacidade para uma palete em cada momento.

Após a execução dos *behaviours* da Figura 24 e da Figura 25, é possível encaminhar um produto desde uma passadeirade chegada até uma passadeira de saída. Estes *behaviours* tornam possível a execução de um comportamento igual ao descrito na Figura 23.

4.1.4. Conhecendo os RA's e CLA's associados

Para que cada HUA saiba que RA's e CLA's estão associados a ele próprio, foi necessário implementar uma interação que é, despoletada pelo TEA, aquando da alocação de RA's ou CLA's

Sendo assim, quando um TEA recebe o *Request*, para que seja acoplada uma nova estação, no espaço físico controlado por este, será enviada uma mensagem para o HUA de entrada, onde é informado que o RA ou CLA com um determinado AID, enviado na comunicação, entrou no sistema e está colocado nesse TEA. Esta informação é de vital importância, pois permite a cada HUA, aquando da chegada de um produto ao seu encaminhador, caso este tenha como etiqueta de destino, esse HUA, a habilidade de saber, através dessa etiqueta, qual o TEA final, que é seu vizinho.

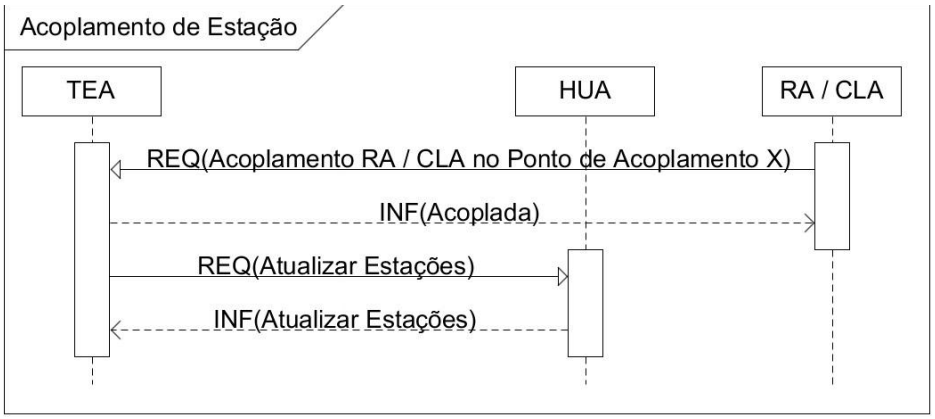


Figura 26 - Diagrama de sequência com a atualização de estações

Como se pode verificar na Figura 26 o HUA necessita de um *behaviour* responsável por receber o pedido de atualização. O AID do RA ou CLA a acoplar é recebido sobre a forma de *String* no *content* da mensagem. Este *behaviour* do tipo *AcheieveREResponder* tem o nome de *ProcessDiverterResourcesUpdateFromConveyor*.

De forma a guardar todos as estações acopladas aos TEA's vizinhos, o HUA guarda em cada TEA da sua BD os RA's e CLA's acoplados.

4.1.5. Definir a Separação entre Camadas

De forma a deixar as camadas das várias partes da implementação, bem separadas, e tornando assim a integração entre os vários sistemas muito mais esquematizada e de fácil resolução, foi implementada uma classede nome *hardware* sobre a forma de *interface*, fazendo com que a comunicação entre a arquitetura implementada e a camada inferior, responsável pela atuação direta com o *hardware*, estivesse bem definida e restrita aos comandos implementados nessa classe.

A Figura 27 ilustra a pilha de camadas que permitem ao HUA atuar e ler no e do *hardware*. Noâmbito deste trabalho foram desenvolvidas as duas camadas de mais alto nível.

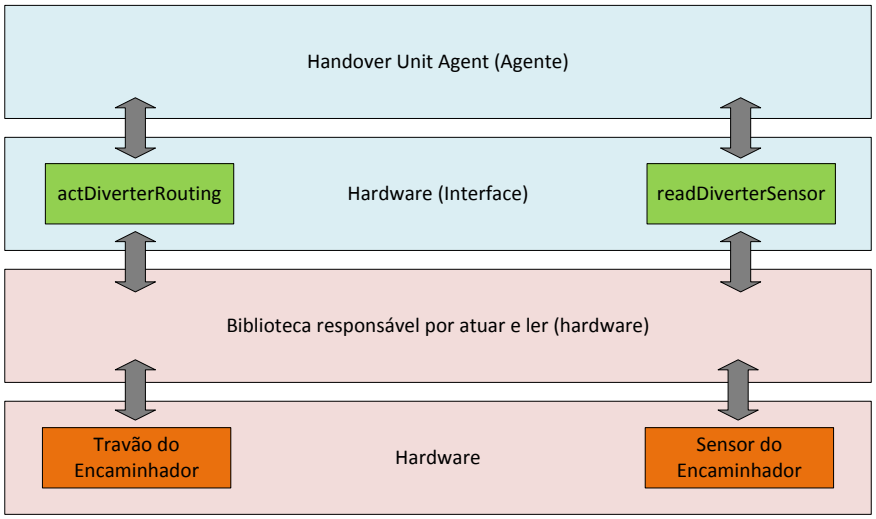


Figura 27 - Pilha de camadas do HUA

A *interface* desenvolvida tem um total de quatro ações, com as seguintes funções:

- *readDiverterSensor*: é responsável por ler do *hardware* o sensor de presença de peça no encaminhador;
- *actDiverterRouting*: é responsável por atuar o encaminhador, tendo em vista o encaminhamento do produto. Recebe como parâmetro de entrada o *ID* da passadeirade destino;
- *initHardware*: Esta função é chamada no início da execução do HUA, tendo em vista a inicialização desta *interface*;

- *updateHardware*: É chamada de forma repetitiva permitindo a atualização dos atuadores. Neste caso específico, quando o encaminhador é atuado existe uma janela de tempo em que este necessita estar na posição para a qual foi deslocado, posteriormente a essa janela de tempo estar cumprida é essencial que o atuador volte à sua posição inicial, este regresso à sua posição inicial é executado por esta função;

Com esta separação de camadas fica facilitada a adaptação aos mais variados sistemas. As alterações necessárias aquando da integração a qualquer sistema, seja ele real ou simulado, restringem-se à classe *Hardware*, configurando-se a maneira como é chamada a escrita e leitura do *hardware.*

4.2. *TransportEntityAgent*

4.2.1. Configuração

Tendo em conta todas as características da passadeira descritas no capítulo da arquitetura, são necessárias várias etapas, todas elas bem distintas, de configuração do TEA.

4.2.1.1. Associação ao Sistema de Transporte

Como foi visto no capítulo anterior o TEA necessita aquando do seu arranque associar-se ao ST onde vai operar.Para a associação de um novo TEA ao ST foi implementado um *behaviour* do tipo *AchieveREInitiator* (Figura 28).

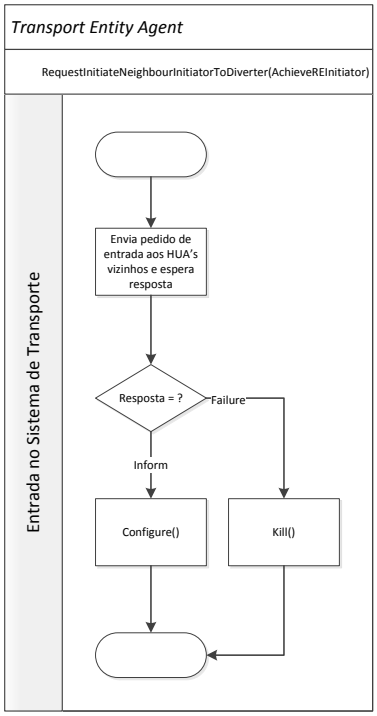


Figura 28 - *Behaviour* responsável por pedir aos vizinhos a entrada

O *behaviour* com o nome *RequestInitiateNeighbourInitiatorToDiverter* inicia um pedido de entrada a ambos os vizinhos, caso alguma das respostas seja negativa este termina a sua execução, não entrando no ST.

4.2.1.2. Acoplamento e Desacoplamento de RA's ou CLA's

Como já foi referenciado, cada passadeira tem pontos na sua estrutura capazes de receber estações, onde poderão ser executados as habilidadessobre os produtos. Esses locais como se sabe têm características próprias, descritas na arquitetura, no que respeita a travões e pré-travões. A cada associação a uma estação de acoplamento, todos os RA's e CLA's que se predispõem a executar as habilidades nessa estação, deverão comunicar a sua entrada ao ST, fazendo-se notar naquele ponto, tornando-se assim visíveis e alcançáveis.

A sequência de interações e mensagens trocadas, pelos elementos presentes nestas comunicações está descrita na Figura 29.

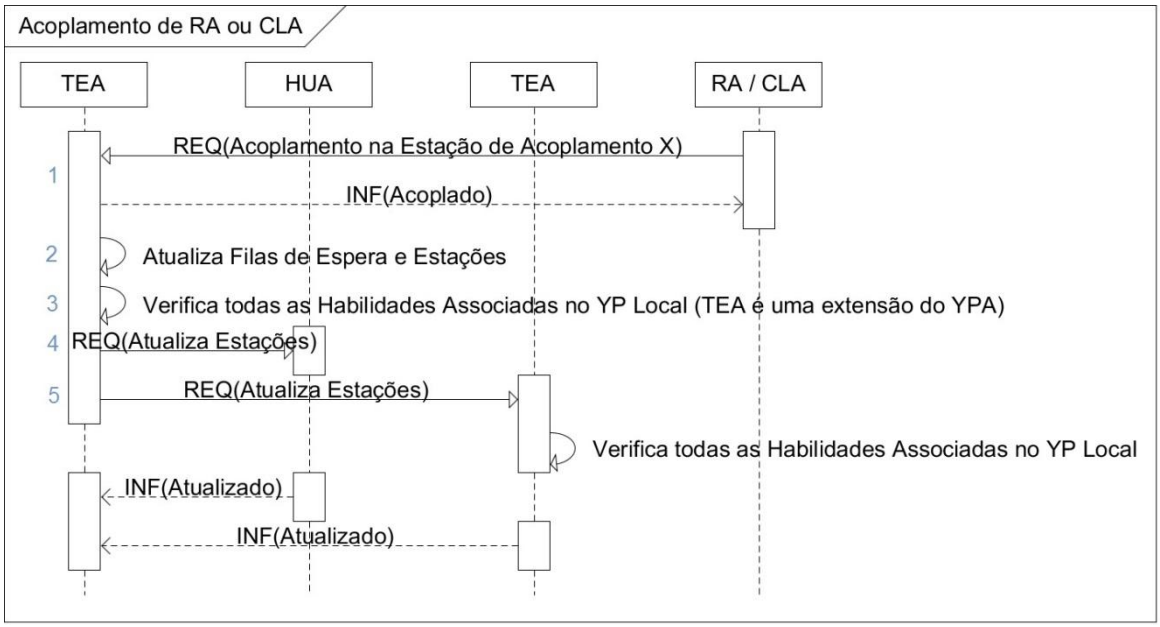


Figura 29 – Diagrama de sequência representativa da evolução do sistema durante o acoplamento de um Recurso ou CLA

Quando o RA ou CLA endereça o pedido de acoplamento ao TEA (ponto 1 da Figura 29), este responde afirmativamente caso não haja nenhum problema com o ponto de acoplamento pedido, mais à frente verificar-se-á quais as limitações referentes à utilização deste valor. Após uma resposta afirmativa por parte do TEA, este atualiza as suas filas de espera e estações presentes na sua passadeira (ponto 2 da Figura 29), e verifica quais as habilidades disponibilizadas por este novo RA ou CLA (ponto 3). A consulta ao YP que permite verificar as habilidades é feita através de uma *query* ao YP local, pois o TEA é uma extensão do YPA.

No ponto 4 do diagrama de sequência está representado o pedidos enviado ao HUA de entrada, para que este saiba que deve atualizar a informação referente às estações deste vizinho. No ponto 5 é enviado um pedido a todos os outros TEA's para que estes atualizem os RA's e CLA's disponíveis no HUA de saída deste TEA. Para completar esta informação todos os TEA's aquando do processamento deste pedido deverão consultar o YP de modo a verificar as habilidades disponíveis pelo novo RA ou CLA acoplado.

Com esta extensão do YPA, o TEA tem a capacidade de fazer pesquisas no serviço de *Yellow Paging*, neste caso para se consultar as habilidades executadas por cada um dos RA's e CLA's.

Tendo como base a Figura 30, que representa o acoplamento da estação. Quando o objetivo é o desacoplamento de um determinado RA ou CLA as diferenças verificadas incidem sobre o facto de o TEA remover o registo da sua BD e comunicar ao HUA de entrada e aos restantes TEA's a saída do ST do mesmo. A consulta ao YP deixa de ser necessária neste caso.

De forma a tornar possível essa característica, implementou-se dois *behaviours* do tipo *AchieveREInitiator*, um de forma a informar todos os HUA's do novo, RA ou CLA, presente e em que TEA se encontra, e outro para informar todos os TEA's.

Para além destes dois *behaviours*, responsáveis pela comunicação, para possibilitar o acoplamento foi necessário implementar um *registerPrepareResultNotification* que arranca um *behaviour*, responsável por verificar o resultado da consulta feita ao YPA, posteriormente processando-a.

No TEA implementou-se um total de sete *behaviours*, responsáveis por processar a entrada e saída, de RA's e CLA's do sistema.

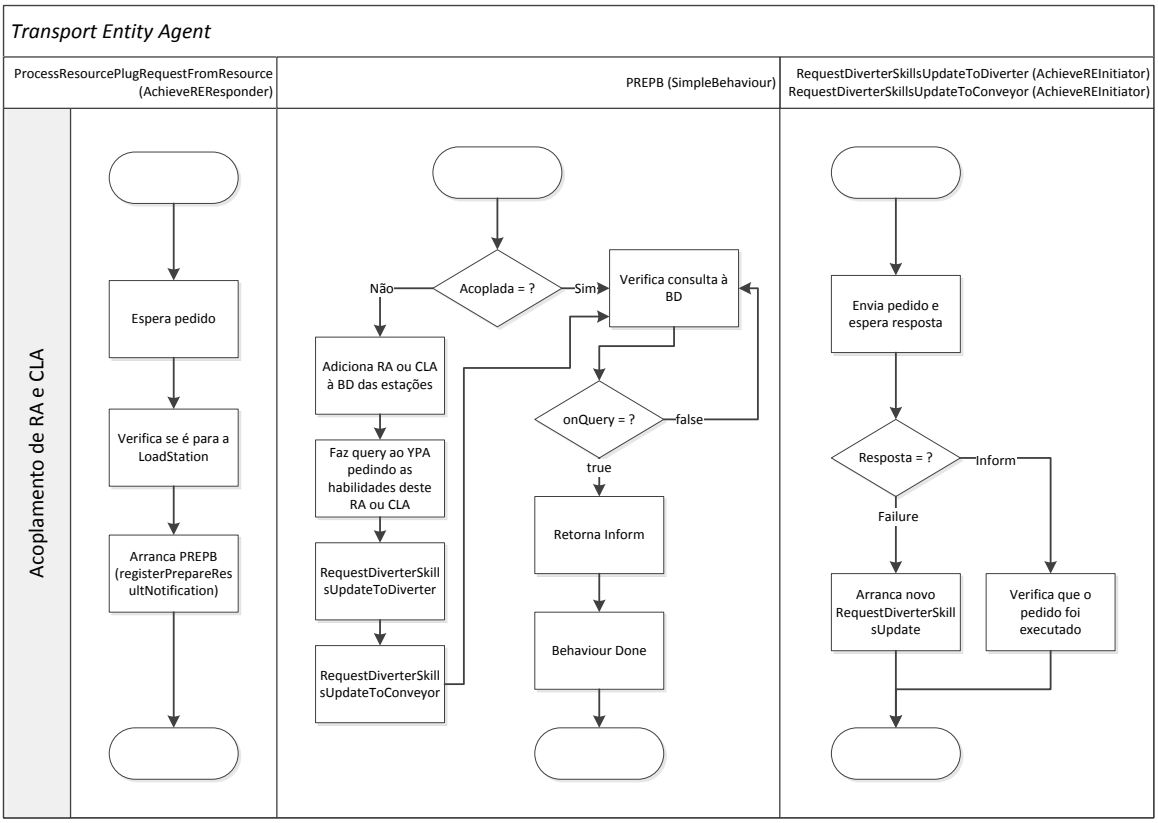


Figura 30 – *Behaviours* responsáveis por tratar do acoplamento de um RA ou CLA

Quando o objetivo é fazer o caminho inverso, o desacoplamento, a sequência de tarefas a ser executada por cada um dos agentes é em tudo muito semelhante, excluindo apenas que cada elemento do ST deixa de consultar o YPA. A estação presente num determinado estação de acoplamento,ao desejar sair do sistema, obrigará a que cada Recurso e CLA presente nessa estação façam um *Request* ao TEA onde a estação está acoplada para que este, respondendo afirmativamente, permita a saída destes agentes sem a existência de problemas.

No caso dos *behaviours* responsáveis pelo desacoplamento e tendo como base de comparação a semelhança aos da Figura 30, verifica-se que o *behaviour* responsável por processar o pedido recebido, *ProcessResourceDesacoplamentoRequestFromResource*, não arranca um *behaviour* para enviar mais tarde a resposta, e chama à sua responsabilidade, a remoção do RA ou CLA e das habilidades associadas a este, e mais tarde inicia *behaviours* do tipo *AchieveREInitiator* pedindo a TEA's e HUA's que removam toda a informação relevante a este RA ou CLA.

A BD responsável por agregar esta informação,está estruturada de uma forma hierárquica, de forma a facilitar a procura na mesma, no diagrama de classes que a representa (Figura 31), pode-se ver em pormenor qual a estrutura utilizada.

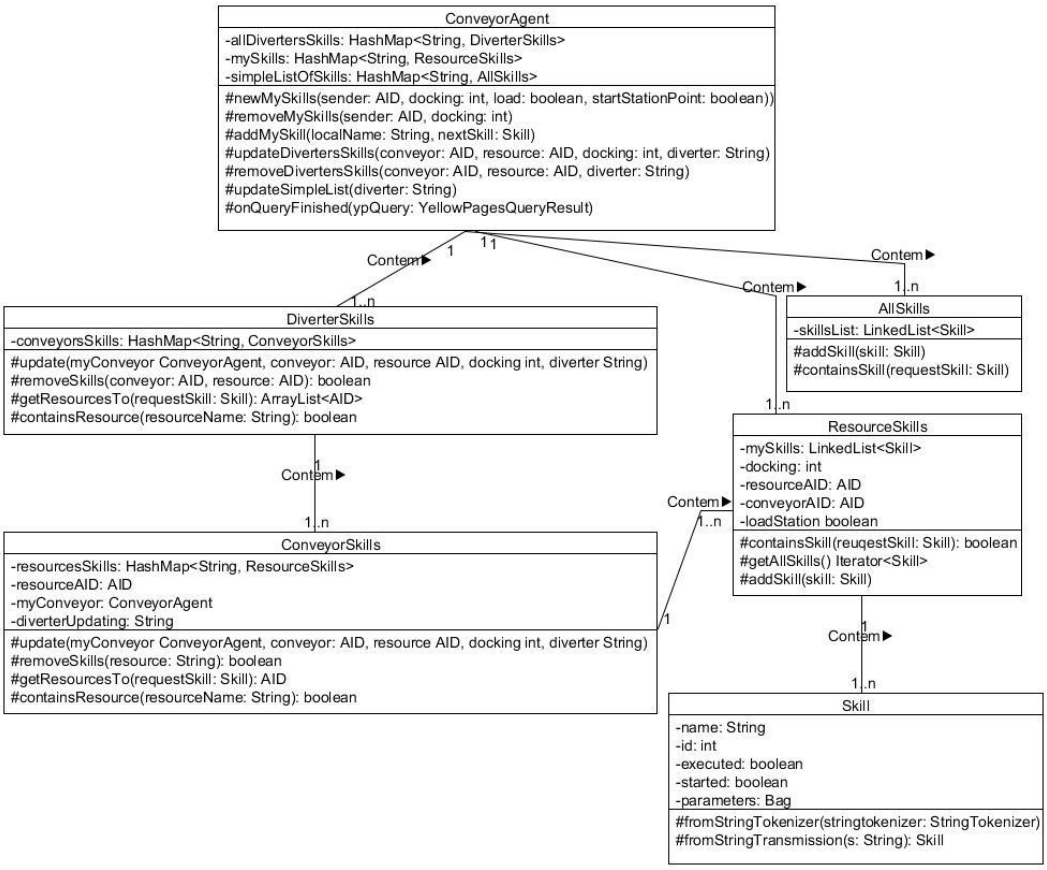


Figura 31 – Diagrama de classes referente à base de dados dos Recursos e CLA's presentes em cada elemento do sistema de transporte

Cada TEA é constituído por uma classe *ConveyorAgent* que faz toda a gestão do agente, com os *behaviours* e as estruturas de dados necessárias à execução do mesmo. A BD *allDivertersSkills* contem a informação referente a cada HUA e aos TEA's vizinhos de uma forma hierárquica. Cada entrada nesta BD tem um objeto do tipo *DiverterSkills*, sendo este constituído por uma lista de *ConveyorSkills*, seus vizinhos. Cada elemento deste tipo tem uma lista de *ResourceSkills* que representam todos os RA's e CLA's acoplados neste vizinho. Com esta informação é assim possível saber que habilidades estão disponíveis em cada um dos RA's, CLA's, TEA's e HUA's de uma forma estruturada e organizada.

Com base na BD *allDiverterSkills*é calculada a *simpleListOfSkills*, este mapeamento apenas é responsável por corresponder o IDde cada HUA a uma lista de habilidades, esta foi criada apenas para facilitar uma posterior consulta.

Para além destas duas tabelas, foi criada a *mySkills*, com os RA's, CLA's e habilidades apenas locais, para uma melhor gestão interna por parte do TEA.

4.2.1.3. Atualização da Estrutura Interna da Passadeira

Após a execução de um acoplamento ou desacoplamento por parte de uma estação, a estrutura de dados, como já foi frisado no capítulo anterior, é alterada de forma a permitir um correto fluxo de produtos entre os vários pontos da passadeira.

De forma a permitir flexibilidade e configurabilidade a quem utiliza a implementação realizada, foram adicionadas algumas variáveis de controlo importantes. Essas variáveis, que serão utilizadas pelo agente TEA, têm como função permitir ao agente, saber quais as características das passadeiras do sistema onde está a correr, neste caso específico, dados referentes à célula da MASMEC. As variáveis de controlo são:

- BOUNDARY_TO_DIVERTERS: é responsável por garantir um espaço mínimo entre o primeiro travão encontrado na passadeirae a sua entrada;
- DISTANCE_TO_PRE_STOPPER: define o espaço entre cada estação e o pré-travão que a antecede;
- USED_SPACE_FROM_STATION: indica o espaço necessário a cada estação;

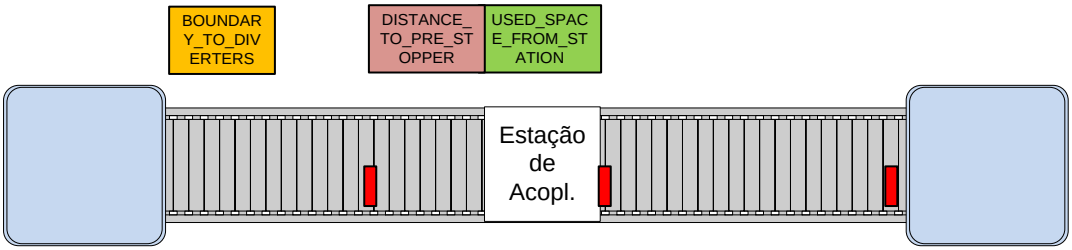


Figura 32 – Representação das variáveis de controlo disponíveis

Todas estas variáveis têm como unidade a palete, por exemplo, se uma delas tiver o valor dois, isso indica que terá uma distância equivalente ao comprimento de duas paletes.

4.2.1.4. Dados Relevantes aos Estados da Passadeira

Durante a execução do TEA, este necessita conhecer um conjunto de informação que lhe indicará o estado da sua passadeira. Essa informação terá de ser ao nível das estações criadas pelo acoplamento dos respetivos RA's e CLA's, da gestão das filas de espera e sub filas de espera, e dos produtos nesses mesmos pontos da passadeira e a sua ordem.

Quando um RA ou CLA pretende efetuar o acoplamentoeste é associado à BD descrita em pormenor no ponto anterior, fazendo um mapeamento entre essa entidade e as habilidades associadas. Posteriormente é feita uma atualização das estações que constituem a passadeira.

O TEA tem na sua BD uma *TreeMap*, cuja função é guardar todas as estações presentes na sua passadeira. Caso a estação já exista são adicionadas mais habilidades, caso contrário é criada uma nova entrada nessa *TreeMap* e posteriormente associadas ashabilidades. De forma a organizar a informação na *TreeMap* recorreu-se a um *comparator*. O *comparator* permite manter a informação da *TreeMap* organizada segundo um critério, descrito pelo *comparator*. Neste caso quando é adicionada uma nova entrada à *TreeMap* esta é reorganizada de forma a manter uma lista ordenada das estações pela ordem de pontos de acoplamento. Esta organização da informação permite uma fácil procura, que se traduz também numa procura mais eficaz da mesma.

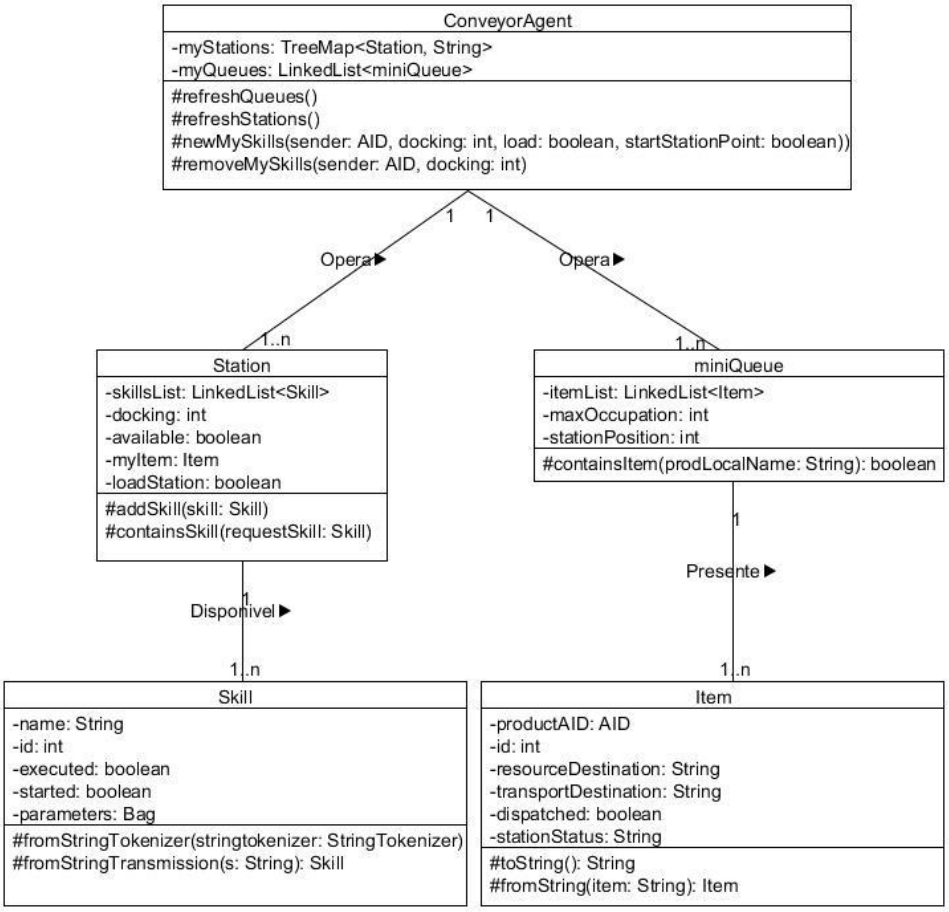


Figura 33 – Diagrama de classes, das classes responsáveis pela gestão de estações e sub filas de espera

Cada entrada na *TreeMap* tem um objeto do tipo *Station*, desenvolvido precisamente para este caso, onde é guardada toda a informação referente à estação, como as habilidades disponíveis, o produto que lá se encontra em cada momento, caso esteja algum, entre outras variáveis de controlo. Estas variáveis de controlo serão explicadas ao pormenor posteriormente.

No caso da gestão das sub filas de espera, agrupou-se a informação numa *LinkedList* em que os elementos constituintes dessa lista são objetos do tipo *miniQueue*. Cada objeto do

tipo *miniQueue* tem na sua constituição a informação relevante à sua gestão, como uma lista de PA's presentes, a sua ocupação máxima e a estação de acoplamento onde está associada a estação exatamente à sua frente.

Cada PA guardado na lista é guardado sobre a forma de *Item*, tendo associado dados referentes às características do PA e também aos vários estados que o PA pode tomar para o ST. Como sendo o caso do destino requerido, ao nível do transporte e dos RA's ou CLA's, como no estado da execução de uma determinada habilidade, ou encaminhamento.

4.2.2. Informação Necessária ao Encaminhamento

Como indicado na arquitetura, o TEA necessita ter em sua posse uma TE, onde baseará as suas decisões, aquando do pedido por parte do PA para que o TEA lhe indique os custos de o encaminhar até uma estação, onde possa ser executado um determinada habilidade.

Para ter acesso a essa tabela, o TEA não a calcula, mas sim utiliza a tabela calculada pelo HUA de saída. Foi então implementado um *behaviour* (Figura 34),que quando iniciado, envia um *Request* ao HUA. A resposta deste deverá conter a sua TE, serializada e devidamente enviada.

Este *behaviour* é inicializado, tanto quanto é atualizado o valor do custo de atravessar estapassadeira. O pedido será enviado para o HUA de entrada, através de um *AchieveREResponder* e o cálculo do valor do custo (Figura 34) foi implementado à custa de um *TickerBehaviour*, que frequentemente atualiza o custo, recorrendo à fórmula da métrica e guardando-o.

É necessário relembrar também, que para as tomadas de decisão alusivas à atribuição de um custo de transporte, para se atingir uma determinada habilidade, para além desta informação é necessário saber que habilidades podem ser operadas nos vizinhos de cada HUA. Essa informação está na posse do TEA, como se pôde ver explicitamente no diagrama de classes para o efeito (Figura 31).

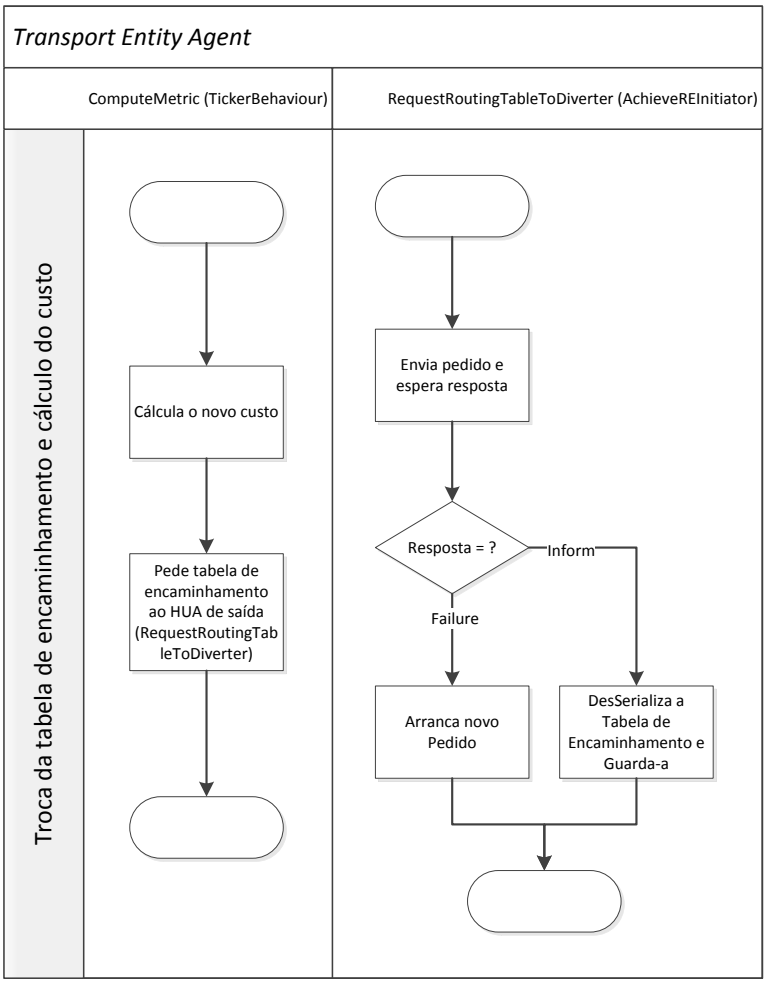


Figura 34 – *Behaviours* responsáveis por pedir a TE ao HUA de saída e atualizar o custo de atravessar a passadeira

Todas as mensagens trocadas pelos agentes são serializadas recorrendo à biblioteca *StringTokenizer*. Esta biblioteca permite que se construa uma *String* contendo diferentes valores, todos eles guardados sobre a forma de *String*. A separação do conteúdo é feita por um conjunto de caracteres definidos pelo desenvolvedor. Foi adotada esta solução devido à elevada carga computacional exigida pelo recurso ao XML. Essa carga é impossível de suportar num ambiente real de produção, onde o processamento é feito por controladores com algumas limitações de *hardware*.

O cálculo do novo custo associado ao TEA, realizado no *behaviourComputeMetric* presente na Figura 34, é baseado na expressão já referida na arquitetura. Nas constantes à disposição foram usados os seguintes valores.

Tabela 3 - Valores atribuidos às constantes presentes na expressão do custo

Constante	*Valor*
a	1
b	1
c	10

Com estes valores a expressão resultante é $Custo = \mathrm{CP} + NE + 10 \times NP$. Rapidamente se percebe que com esta atribuição de valores a preponderancia reside no número de produtos presentes na passadeira. Esta preponderancia tem como objetivo o balanceamento de carga entre as várias passadeiras.

4.2.3. Execução do Produto

Durante a sua execução, o PA toma um conjunto de estados, consoante a fase da execução. Para controlar estas fases, e de forma ao ST conhecer constantemente qual a fase em que o PA se encontra, foi adicionada uma variável de estado, que pode tomar os seguintes valores:

- ITEM_IN_APPROACH: indica que um produto entrou no espaço da estação, mas ainda se encontra a caminho, este valor permite à estação saber que tem um PA prestes a operar, mas ainda não chegou ao local preciso da execução;

- ITEM_EXECUTING_SKILL: durante a execução de uma habilidade sobre o PA, o TEA necessita saber que este não pode sair da estação;

- ITEM_READY_TO_DISPATCH: neste estado o PA já executou a habilidade pretendida e está à espera para ser despachado;

- ITEM_IN_ROUTING: o PA ainda não está na estação de destino, encontra-se em encaminhamento;

- ITEM_IN_LOAD: este estado serve para indicar que a palete não tem nenhum PA associado. Este caso surge quando a palete é adicionada ao sistema, ou quando esta depois do PA sair fica em espera, vazia, na estação de entrada;

Com estes valores que a variável de estado pode assumir, é possível cobrir todas as hipóteses para o PA.

Durante a execução do PA, este comunica com o transporte em três ocasiões, sendo elas, o pedido de uma lista de RA's e/ou CLA's que podem executar uma determinada habilidade e o custo ao nível do transporte, o pedido de encaminhamento por parte do PA, quando este já

tomou uma decisão quanto ao destino que pretende, e quando o ST envia um *Inform* ao PA indicando-lhe que este já se encontra na estação de destino e que pode pedir a execução da habilidade.

A entrada do produto no ST, ou seja, a associação deste a uma palete vazia, é feita pelo pedido de um custo para a habilidade de entrada, assim o ST adiciona o PA à palete vazia. Posteriormente, e para cumprir o protocolo desenvolvido, o PA pede para ser encaminhado até à estação de entrada, sendo prontamente enviado o *Inform*, indicando a chegada ao destino.

No final da execução, o PA pede que seja encaminhado até uma habilidade vazia, ou seja, ao pedir para encaminhar para uma determinada habilidade, este campo vem vazio, sendo processado pelo ST como um pedido para sair, cumprindo-se assim o protocolo.

A sequência da execução é descrita pelo diagrama que se segue (Figura 35), onde a azul está indicado o valor da variável de estado do produto.

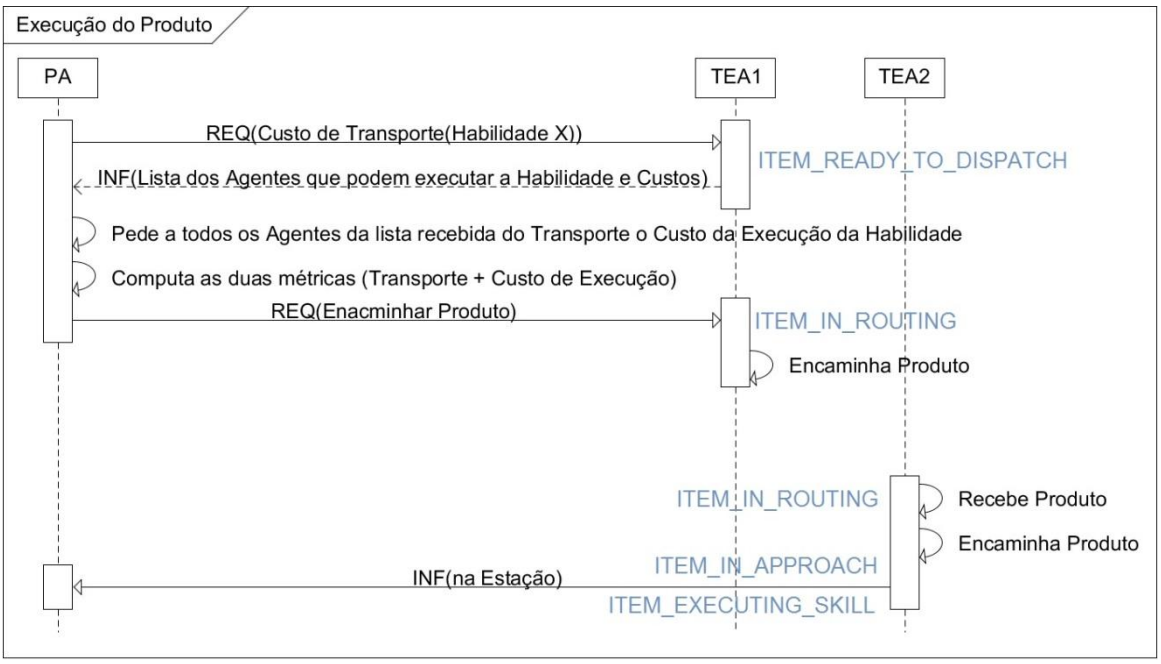


Figura 35 – Diagrama de sequência das interacções do Produto com o Sistema de Transporte durante a sua execução

De forma a permitir estas interações, foram desenvolvidos dois *behaviours* de receção de mensagens, do tipo *AchieveREResponder*, para responder aos pedidos, dos custos e de encaminhamento.

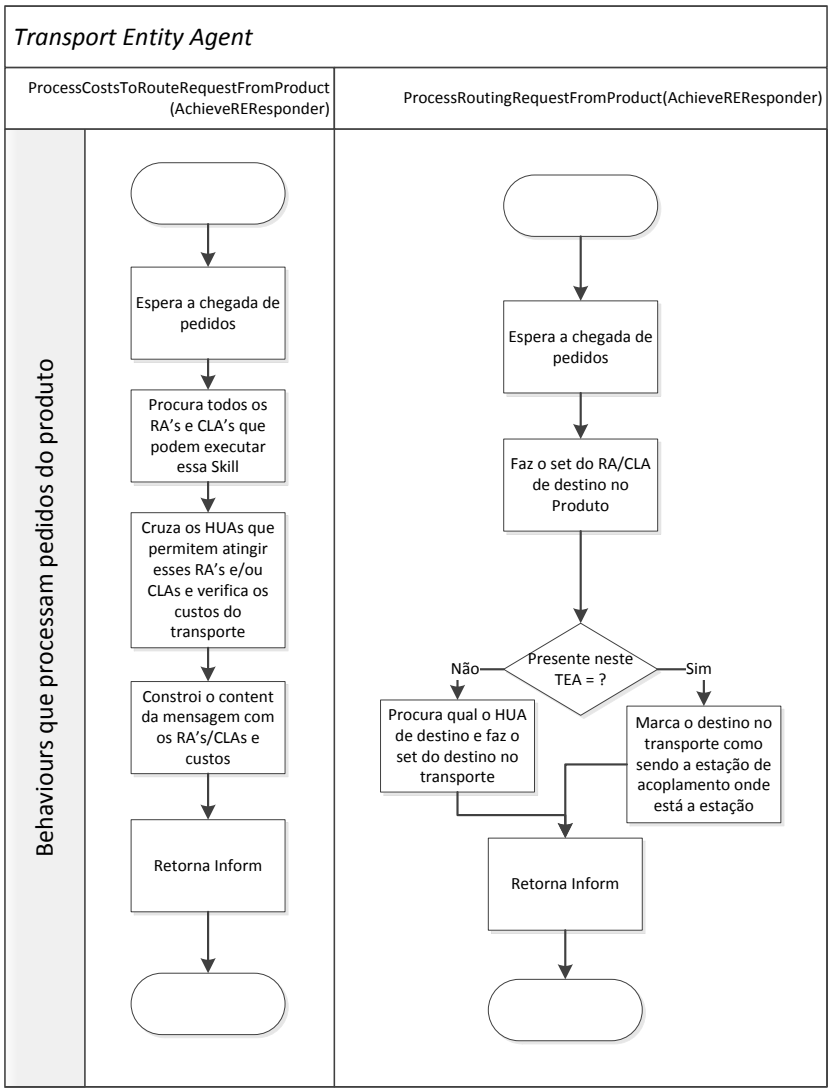


Figura 36 – *Behaviours* responsáveis por processar os pedidos do Produto

Com estes dois comportamentos (Figura 36), fica apenas em falta a última comunicação, que acontece quando o TEA informa que o PA chegou ao seu destino, para essa característica não se recorreu a nenhum *behaviour*, pois o envio da mensagem é feito diretamente sem recorrer a um novo *behaviour*.

Para além destas trocas de mensagens com o Produto, o TEA necessita de mais algumas características implementadas, tendo em vista o correto encaminhamento do produto até ao seu destino. Essas características vão permitir, como foi descrito na arquitetura, encaminhar o Produto ao longo da Passadeira, avançando de sub fila de espera para estação e de estação para *s*ub fila de espera, e posteriormente quando este se encontrar no final da passadeira encaminhá-lo para o HUA de saída.

4.2.4. Renovação do Estado da Passadeira

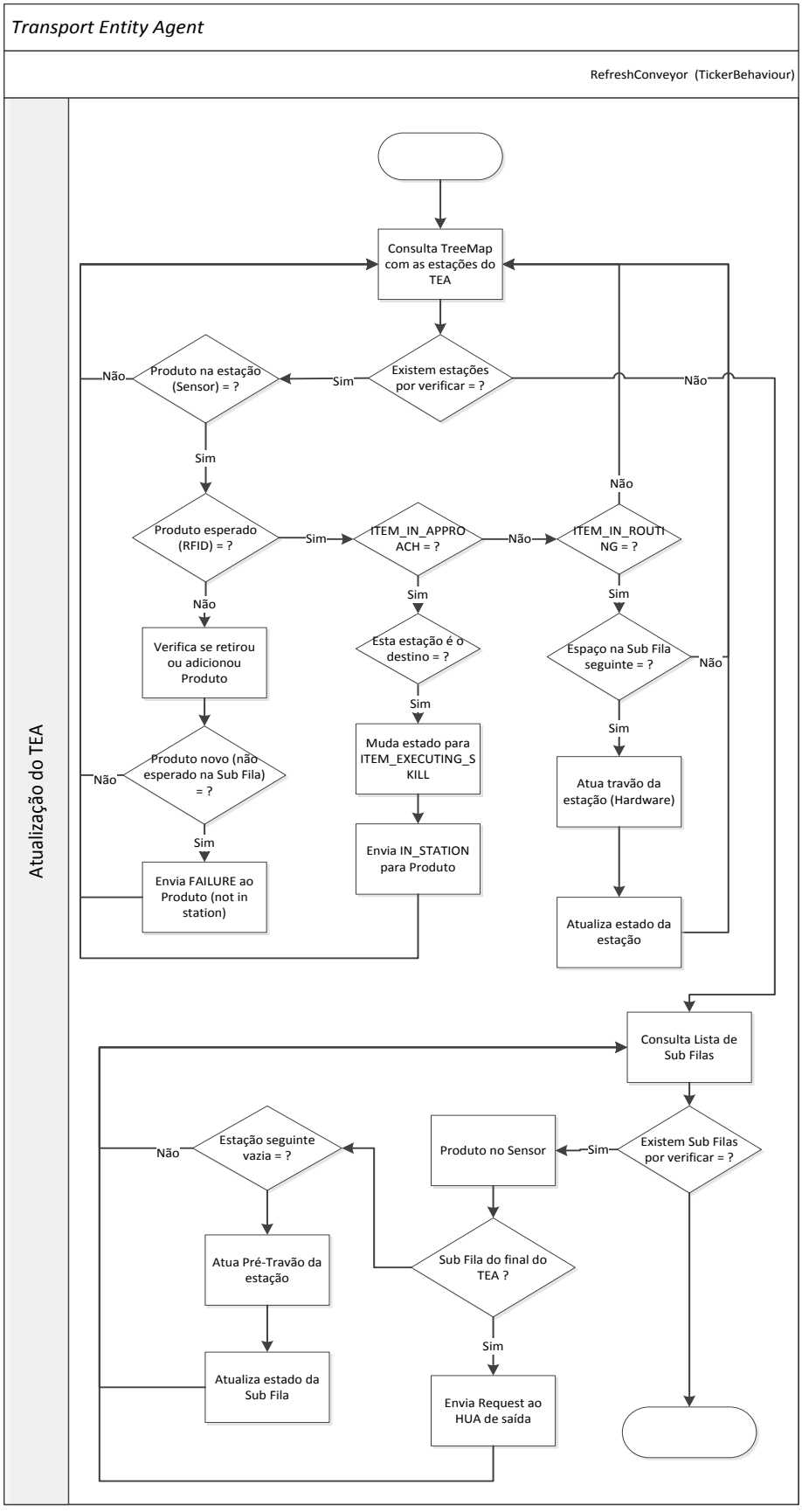


Figura 37–*Behaviour* que executa a actualização do estado do *Passadeira*

O *behaviour* responsável pela renovação da informação (Figura 37) referente à passadeira, e das tomadas de decisão relativas à atuação dos travões, mandando avançar uma palete, foi implementado recorrendo a um *TickerBehaviour*, que assim obriga o TEA a executar esta rotina de um em um segundo. Este intervalo foi ajustado de modo a ser o mais pequeno possível, devido à necessidade de rapidamente se encaminhar os produtos. Para além desta necessidade é importante ter em conta o tempo de atraso nas chamadas ao *hardware*. O diminuo do intervalo entre *tickers* resulta também num aumento do processamento, devido ao aumento de número de vezes que esta verificação é efetuada.

Este *behaviour* para além de mandar avançar as paletes é também responsável por verificar a ordem de chegada dos produtos a cada estação. A verificação é feita recorrendo à leitura da *tag* de RFID, com esta informação o TEA sabe qual o PA associado a uma determinada palete. No caso de o PA esperado não corresponder à informação lida o TEA verifica a existência deste na sub fila que precede a estação. Caso o PA esteja ainda associado à sub fila foi adicionada uma palete. Caso contrário a palete com o PA esperado foi retirada e assim o próximo elemento da sub fila coincide com o PA presente na estação. Em ambos os casos o TEA reorganizará a sub fila e a informação relativa à estação de modo a se atualizar.

Quando se dá a associação de uma nova palete a uma sub fila, esta deverá ter um PA associado. O TEA ao detetar a entrada de uma nova palete inicia um pedido ao PA onde o questionará sobre o seu destino. Quando o TEA recebe esta informação torna a encaminhá-lo até ao seu destino.

Para um correto fluxo de produtos na passadeira, estão só a faltar os *behaviours* responsáveis por permitir a entrada e o despacho, de e para o HUA. Os *behaviours* para esse efeito são os presentes na Figura 38.

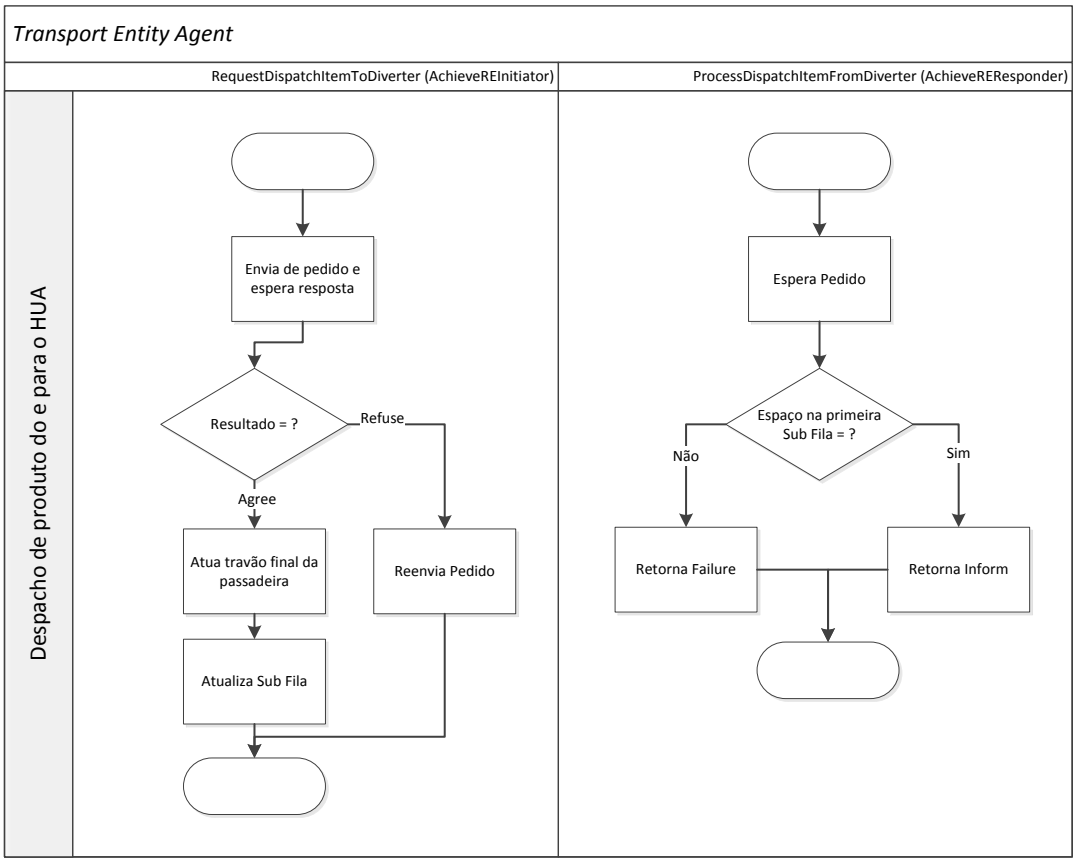


Figura 38–*Behaviours* que permitem o despacho e a recepção de Produto

4.2.5. Entrada de Palete no Sistema

No sistema para onde foi desenvolvida a aplicação, no âmbito do projeto IDEAS, a célula desenvolvida pela MASMEC, não se recorre a um ponto de entrada de produtos e um de saída, de facto o trajeto descrito pelos produtos é feito de forma circular, sem se dar entrada e saída no sistema da palete que o transporta.

Tendo em conta esta característica própria do sistema, em que foi desenvolvido o trabalho, recorreu-se à implementação de estações com características especiais, as estações de *load*. Nestas estações de *load*, o sistema dá a possibilidade, quando estas não se encontram ocupadas adicionar outra palete transportadora. O utilizador introduz uma palete no sistema, por intermédio da *interface* de *deployment*, presente no computador pessoal. Depois de introduzida a palete no ST e esta ter sido colocada fisicamente na estação, o ST está apto a que um produto peça para entrar no sistema através daquela estação, ou seja, naquele ponto do ST. A remoção de paletes pode ser feitaa qualquer momento, pois esta remoção é detetada pelo sistema, por intermédio das verificações feitas através da consulta à informação gravada nas *tags* de RFID.

Quando o PA termina a sua execução, a palete que o transporta não tem ponto de saída, como já foi referenciado. Para cumprir o objetivo, de que a linha continua a processar PA's sempre que possível, a palete depois de vazia, por o *unload* ter sido já executado numa das

outras estações, o ST autonomamente encaminha a palete desocupada para a estação de *load,* para que outro PA possa ser associado a esta palete, sem que seja necessário adicionar mais uma palete ao sistema, visto que esta encontra-se disponível.

4.2.6. Definir a Separação entre Camadas

À semelhança do realizado para o HUA foi implementada uma interface, para permitir uma separação bem definida entre camadas, neste caso entre os agentes do tipo TEA e a camada de *hardware*.

A Figura 39 ilustra a pilha de camadas que permitem ao TEA atuar e ler no e do *hardware*. No âmbito deste trabalho foram desenvolvidas as duas camadas de mais alto nível.

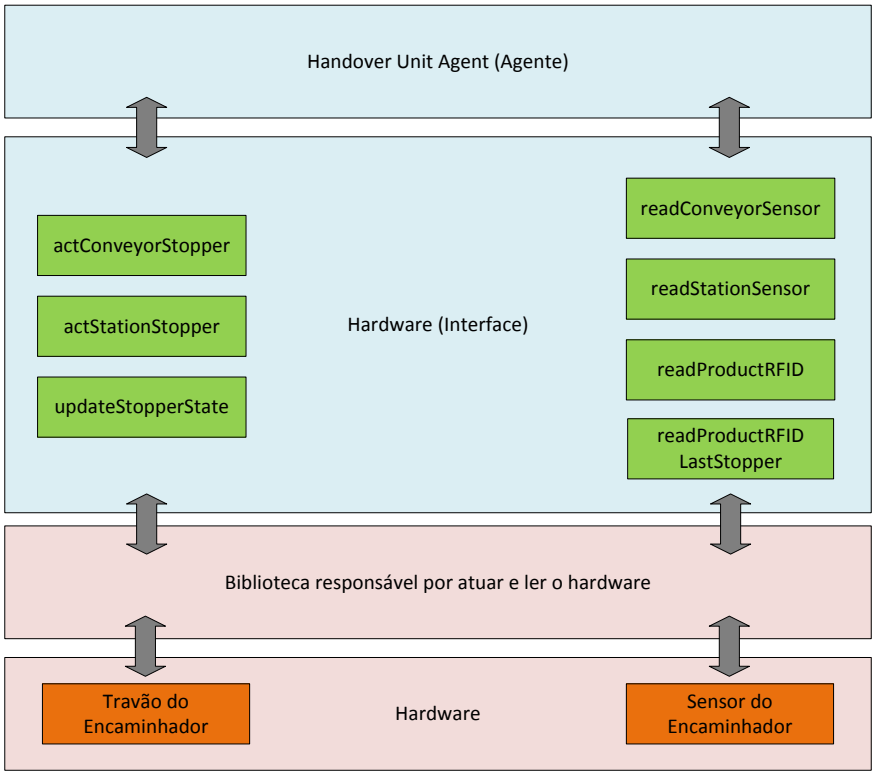


Figura 39 - Pilha de camadas do TEA

Esta interface tem como possíveis chamadas:

- *readStationSensor*: tem como responsabilidade informar da presença numa determinada estação ou no pré-travão dessa estação, recebendo como parâmetro de entrada o estação de acoplamento e um booleano indicando se é o sensor do pré-travão ou do travão;
- *readConveyorSensor*: faz a leitura do sensor presente no final da passadeira, informando da presença de peça antes do travão final;

- *actStationStopper*: faz a atuação do travão ou pré-travão de uma determinada estação, a escolha é feita à semelhança do que é feito com a leitura dos sensores da estação;

- *actConveyorStopper*: atua no travão final da passadeira;

- *initiHardware*: é responsável por inicializar a classe hardware em cada agente como a criação e atribuição dos dados relevantes, e também a inicialização do hardware;

- *updateStoppersState*: os travões quando atuados necessitam de permanecer algum tempo na posição de atuados, permitindo assim que a totalidade da palete com o PA;

- *readProductRFID*: faz a leitura do ID do Produto presente numa determinada estação, ID esse gravado na *tag* de RFID;

- *readProductRFIDlastStopper*: à semelhança do anterior, consulta o ID do PA gravado na *tag*, mas só que neste caso é verificado no final da passadeira;

Com os pontos já referidos, fica descrita a implementação necessária ao correto funcionamento do TEA, desde a configuração do mesmo, as interações com o PA e a correta atualização dos vários estados de execução.

5. Validação e Testes

5.1. Sistema Experimental

Foi também projetada uma célula industrial no âmbito do projeto IDEAS. A célula projetada foi construída pela MASMEC (http://www.ideas-project.eu), especialista em automação industrial.

5.1.1. Sistema Experimental da MASMEC

A célula desenvolvida, Figura 40, é constituída por uma rede de passadeiras,ligadas entre elas por encaminhadores, que executam a passagem de paletes entre as várias passadeiras, operando as decisões no que respeita ao encaminhamento das paletes.

Figura 40 – Sistema experimental da *Masmec*

Este sistema tem a capacidade de acoplar nove estações, todas elas posicionadas em localizações distintas das passadeiras. Um conjunto de seis passadeiras e quatro encaminhadores definem a topologia da rede.

Na célula em estudo o ST é lançado e executado em dois controladores industriais disponíveis para esse efeito, protótipos desenvolvidos também pela MASMEC. Nestes controladores são executados os agentes responsáveis pelo ST.

De forma a incrementar o desempenho do sistema, os agentes do ST foram distribuídos pelos controladores de forma a estes serem executados no controlador onde está a biblioteca de *hardware* na qual vão atuar. A distribuição feita pelos controladores está representada na Tabela 4, onde os controladores são diferenciados pelo seu IP na rede. Os agentes em que o nome se inicia por "D_" abstraiem encaminhadores enquanto os iniciados por "C_" abstraiem passadeiras.

Tabela 4 - Distribuição dos agentes pelos controladores industriais

Controlador 192.168.1.101	***Controlador 192.168.1.102***
D_1	D_3
D_2	D_4
C_1	C_4
C_2	C_5
C_3	C_6

De forma a possibilitar o lançamento dos agentes transporte nos controladores anteriormente referenciados foi necessário desenvolver uma aplicação e dois tipos de agentes responsáveis pelo *deploy* do ST.

Primeiramente foi desenvolvida uma ferramenta responsável por lançar os agentes do tipo transporte, com as suas caraterísticas e vizinhos. Posteriormente foi desenvolvido um agente com o objetivo de este correr no controlador. Este agente ao receber mensagens do agente responsável pela aplicação tem a capacidade de lançar o agente no controlador onde se encontra, tornando assim possível lançar agentes do ST nos controladores.

Isto leva à obrigatoriedade de uma inicialização de todos os agentes de *deploy,* em cada um dos controladores, anterior ao lançamento de qualquer agente de transporte. Neste caso os controladores tinham à sua disposição uma consola de *Linux*, fazendo com que o lançamento destes agentes de *deploy* fosse feito pelo recurso a essa mesma consola. Lançados estes agentes de *deploy* o utilizador pode assim iniciar a configuração do ST através da aplicação desenvolvida para o efeito, onde um agente envia os pedidos de inicialização a cada um dos agentes de *deploy* pretendidos.

Na Figura 41 está representado um diagrama que representa o sistema de *hardware* disponível e como se ligam.

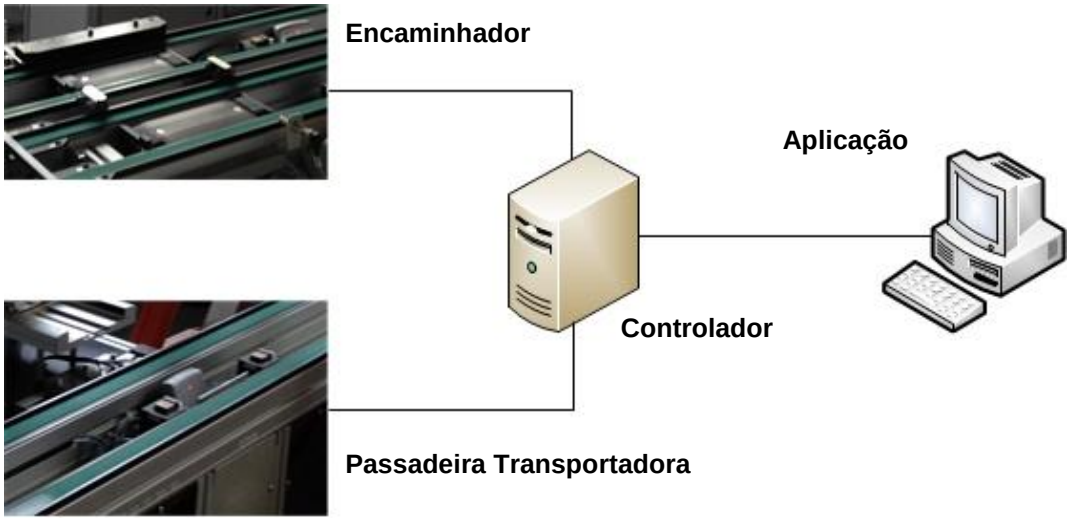


Figura 41 – Esquema representativo dos componentes de *hardware*

Deste modo é possível lançar todo o ST através de uma aplicação de forma fácil e prática. Com esta implementação o utilizador pode também a qualquer momento adicionar novos elementos a um sistema desde que no controlador de destino esteja já em execução um agente de *deploy*.

5.1.2. Execução do Sistema

De forma a tornar possível a execução do sistema apenas teve de ser desenvolvida a classe *hardware* que define a interface entre os agentes e as chamadas ao *hardware*, interface essa desenvolvida para esse efeito.

Os testes possibilitaram verificar o correto funcionamento da arquitetura projetada num sistema real. Foram executados alguns testes de modo verificar o comportamento do sistema abstraído pelos agentes transporte:

- Testes de carga;
- Acoplamento de estações;
- Desacoplamento de estações;

Com estes testes foi possível verificar o correto funcionamento do sistema e as suas principais caraterísticas.

5.2. Sistema Virtual

De forma a testar algumas situações interessantes no ST implementado recorreu-se a um ambiente virtual.

5.2.1. Caso de Teste e Simulação do Sistema

A arquitetura apresentada foi testada, para a finalidade da sua validação, em simulação. De modo a tornar a simulação o mais realista possível, os agentes utilizados são os mesmos que os utilizados para controlar o sistema real detalhado na arquitetura do IADE (Figura 1).

Estes agentes estão ligados a um simulador baseado em agentes que garante o transporte virtual de paletes em todo um ambiente virtualizado. De um ponto de vista arquitetural, a integração que permite esta resolução está descrita na arquitetura do IADE.

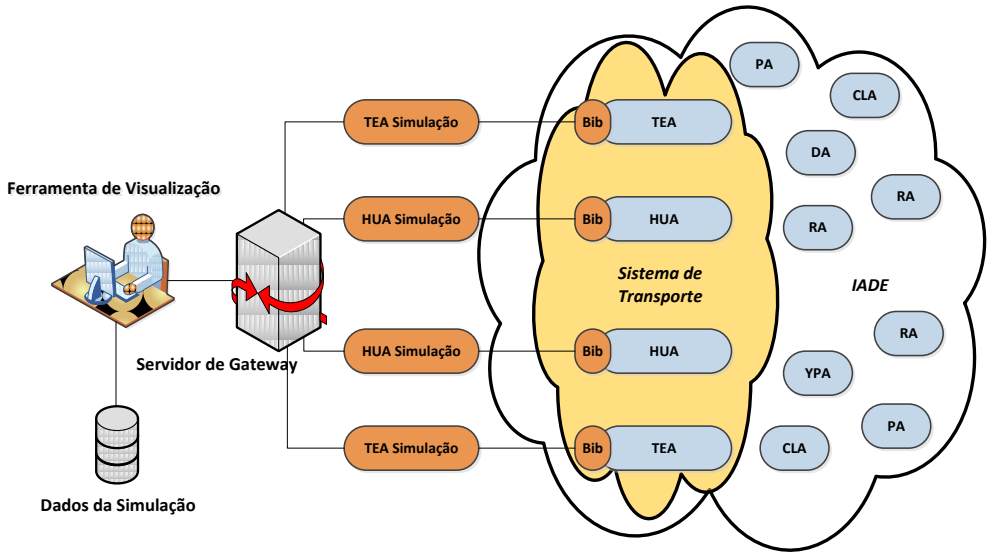


Figura 42 – Visão conceptual da integração entre a plataforma e o ambiente de simulação

A ferramenta de simulação desenvolvida no âmbito de outro trabalho de Mestrado,é baseada em eventos e incluiu um agente central, que controla o referencial do tempo em todos os agentes simulação, que na imagem estão a cor-de-laranja. Cada agente de simulação replica o comportamento de uma passadeira ou um encaminhador. Neste contexto, cada agente possui a habilidade de simular os travões respetivos, mas virtualmente, e os respetivos sensores de presença. Estas características, como já foi descrito, são exigidas pelo TEA e também pelo HUA.

Os agentes pertencentes ao IADE implementam uma biblioteca, também representada a laranja na Figura 42, que permite a entrada do valor referente ao sensor de presença, como a atuação referente aos travões. Esta replicação permite simular um cenário real, na totalidade.

Neste contexto e com estas funcionalidades, todas as ações acontecem de acordo com o sistema real, sendo apenas simulado e virtualizado o deslocamento das paletes e a execução de habilidades.

O sistema utilizado para os testes está representado naFigura 43.

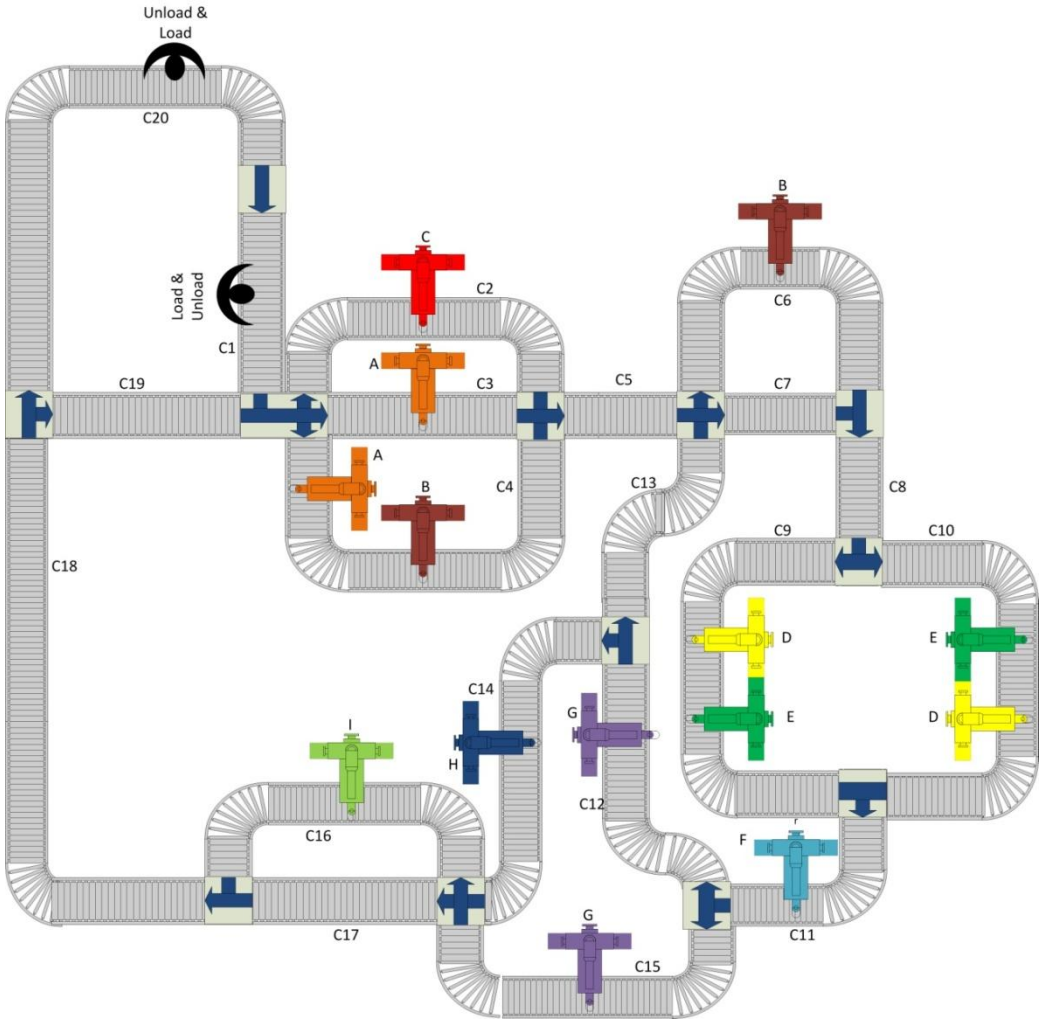


Figura 43 – Sistema utilizado nos testes virtuais

O sistema descrito na figura inclui várias estações e uma rede de transporte, constituída por passadeiras e encaminhadores, complexa. Cada estação (que tem associado um agente RA ou um agente CLA),tem a capacidade de executar uma única habilidade que neste caso está representado por uma letra e uma cor, como por exemplo na passadeira C2 é executada ahabilidade C, e todas as estações com a capacidade de executar ahabilidade C apareceram a vermelho.

A direção para onde os encaminhadores têm a possibilidade de direcionar as paletes está representada pelas setas a azul, no interior da área delimitada por cada encaminhador. A partir de cada ponto do sistema, o sistema tem a possibilidade de oferecer ao PA um conjunto

variado de alternativas, para o encaminhamento do mesmo, fazendo com que o PA tenha várias alternativas para cumprir o seu plano de execução.

O sistema é composto também por duas estações especiais, responsáveis pela introdução e remoção de PA's, do sistema. Cada passadeira tem um ID começado por C, e depois é adicionado um inteiro, único de cada passadeira, este inteiro é também o identificador do TEA que abstrai a passadeira. No que respeita aos HUA's, estes são representados na figura pelos blocos que se encontram entre as passadeiras, delimitando o início e o fim das mesmas.

Nesta simulação foram utilizados um total de oitenta e nove agentes pertencentes à arquitetura do IADE, estando estes apresentados na Tabela 5. O número total de agentes requeridos na execução desta simulação é na verdade bastante superior a este, devido à existência de um agente simulação para cada um dos agentes da arquitetura.

Tabela 5 – Número de agentes IADE, utilizados na simulação, por cada uma das classes

Classe	***Número de Agentes***
Deployment Agent (DA)	1
Estações (RA ouCLA)	14
ProductAgent (PA)	40
ProductSourceAgent (PSoA)	1
ProductSinkAgent (PSiA)	1
TransportEntityAgent(TEA)	20
HandoverUnitAgent (HUA)	12

Cada HUA e estação podem lidar em cada momento com um máximo de um PA, enquanto cada TEA tem um comprimento variável e pode, portanto, lidar com vários PA's simultaneamente, se assim for necessário. O comprimento de cada TEA está representado na Tabela 6. Outra característica importante neste cenário é o facto de todos os TEA's transportarem à mesma velocidade, e esta ser constante durante toda a execução.

Tabela 6 - Comprimento de cada passadeira, em número de paletes

ID da Passadeira	*Tamanho*
C1	8
C2	8
C3	8
C4	14
C5	5
C6	8
C7	5
C8	5
C9	14
C10	14
C11	8
C12	8
C13	5
C14	8
C15	8
C16	8
C17	5
C18	28
C19	5
C20	45

Como mencionado na arquitetura do ST, se as restrições impostas, tendo em vista o correto funcionamento do mesmo, forem cumpridas, teoricamente, e como se está na presença de um sistema virtual, onde não existe limitações físicas, poderá ser criado em qualquer ponto uma estação de acoplamento no ST.

Neste caso de estudo, as posições consideradas para o acoplamento das respetivas estações está detalhado na Tabela 7.

Tabela 7 - Estações com o respectivo *skill* e localização na passadeira

Identificador da Estação	*Skill*	*Passadeira*	*Estação de Acoplamento (Em número de paletes)*
1	Entrada/Saída	1	2
2	Entrada/Saída	20	41
3	C	2	6
4	A	3	6
5	A	4	6
6	B	4	12
7	B	6	6
8	D	9	6
9	E	9	12
10	E	10	6
11	D	10	12
12	F	11	16
13	G	15	6
14	G	12	6
15	H	14	6
16	I	16	6

Para além destas especificações, referentes ao ST, foram utilizados três tipos de PA's. Cada um dos tipos tem um plano de execução distinto, sendo os planos os representados na Tabela 8.

Tabela 8 - Produtos a serem testados no caso de estudo

Product Agent (PA)	*Plano de Execução*
Tipo 1	A,B,C,D,E,F,G,H,I
Tipo 2	A,B,C,D,E,F,G,I
Tipo 3	A,B,C,D,I

Cada PA entra no sistema assim que for possível, ou seja, um determinado PA é criado e adicionado ao sistema, pela estação de entrada, quando esta processar o pedido do PA e iniciar o encaminhamento do mesmo, esta passa a estar vazia, e logo habilitada à entrada de um novo PA no sistema.

Os PA's são misturados em proporção de 1:1:1, resultando no final num igual número de PA's processados de cada tipo. A métrica considerada, é $Custo = \mathrm{CP} + NE + 10 \times NP$, de forma a ter como fator mais influente, o número de paletes em cada passadeira.

5.3. Discussão de Resultados

O cenário anteriormente descrito foi testado vinte vezes, e cada passadeira tem nas figuras abaixo, as vinte amostragens representadas com diferentes cores. O sistema estive em funcionamento durante um total de quarenta minutos (tempo total de produção), durante o qual, foram injetados progressivamente PA's, conforme foi detalhado anteriormente.

A taxa de despacho de paletes por minuto (*throughput*), o número de paletes em cada momento da execução, e os períodos onde ocorre acumulação de paletes, são representados nos dados referentes à simulação. Os gráficos das passadeiras com menor preponderância nesta análise não vão ser apresentados.

O processo, para os três tipos de PA's, começa na passadeira C1, onde são introduzidos todos os quarenta PA's na linha, isto durante aproximadamente seis minutos. Esta passadeira nunca é revisitada por qualquer um dos PA's, visto que durante a sua execução, a recirculação no ST nunca será efetuada por este ramo, e no final da mesma, estes serão encaminhados para a passadeira exatamente anterior a esta, a passadeira C20.

A análise às passadeiras iniciar-se-á pelas C2 (Figura 44), C3 (Figura 45) e C4 (Figura 46).

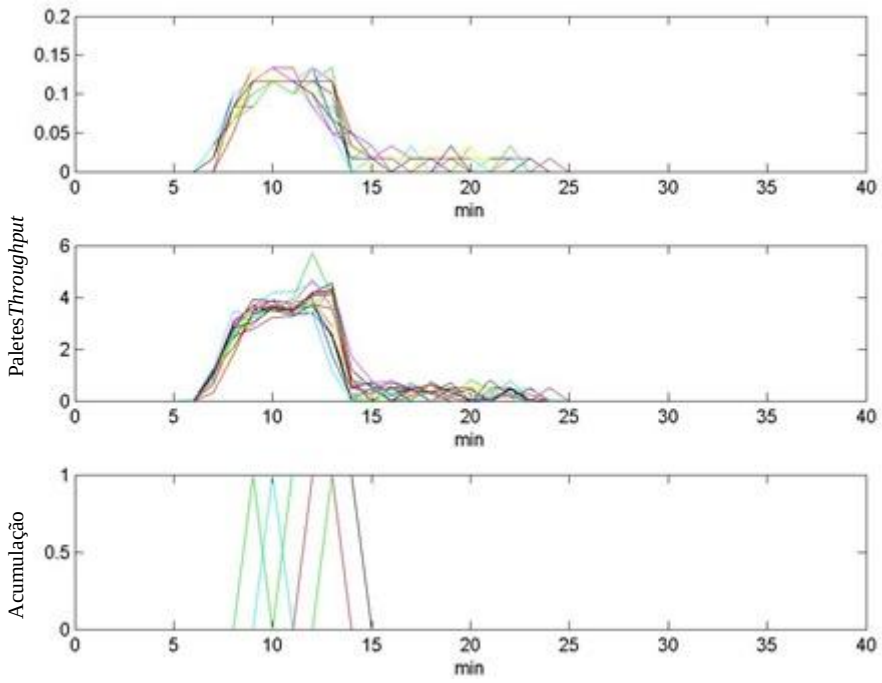


Figura 44 – Comportamento da passadeira C2

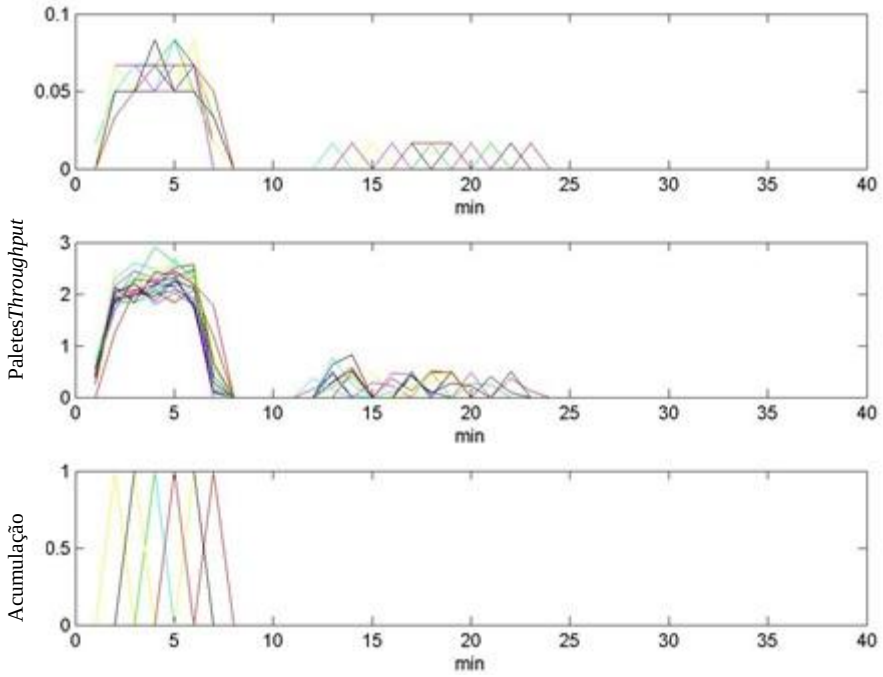


Figura 45 – Comportamento da passadeira C3

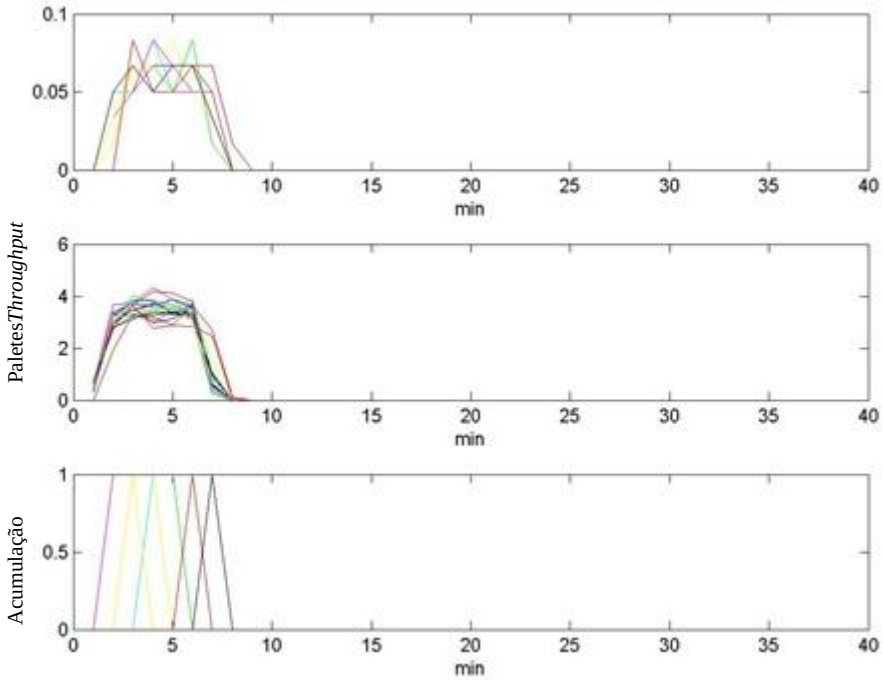


Figura 46 - Comportamento da passadeira C4

Nestas primeiras três passadeiras é possível verificar, que o efeito de recirculação da palete inicia-se após dez minutos desde o início da execução. Este efeito deve-se ao plano de processamento do PA e à localização das estações no sistema. Os locais escolhidos para acoplamento das estações, foram selecionados com o intuito de explorar potenciais problemas de miopia, resultantes das decisões do PA, baseadas no algoritmo atual, para que em futuros desenvolvimentos o PA simulado possa tomar as suas decisões com base num horizonte de

decisão superior a uma estação. No contexto atual, a recirculação deve-se à necessidade de execução da habilidade C, implementada apenas pela estação presente na passadeira C2. Esta recirculação é independente de quaisquer métricas e afeta qualquer PA. O efeito de recirculação verifica-se também no caso da execução dashabilidadesD e E, presentes nas passadeiras C9 e C10, e na execução das habilidadesG e H, onde são operadas, no caso do G nas passadeiras C12 e C15, enquanto a H é operada na passadeira C14. Nestes casos, e na ausência de miopia, a decisão do PA (do tipo 1 e 2) deveria passar por preferir a passadeira C9 para a execução sequencial das habilidades D e posteriormente E, em vez de obrigar o sistema a recirculá-lo mais uma vez, caso este execute ahabilidade D na passadeira C10. O cenário que tem pior desempenho, nesta questão da recirculação, acontece quando o PA, para alem, de pedir a execução dahabilidade D na passadeira C10, executa a habilidade G na passadeira C15., neste caso o PA é submetido a um total de três recirculações. Os PA's do tipo 3 não são afetados, em qualquer dos casos, por estas estações, devido ao curto plano de execução. É também de elevado interesse referir que a C4 não foi atravessada durante a recirculação de paletes, isto deve-se ao impacto da métrica nas decisões, pois com o $Custo = \mathrm{CP} + NE + 10 \times NP$, resulta num total de 16 para C4, 9 para C2 e C3, todos eles sem a presença de qualquer palete. Uma vez que nenhuma das passadeiras com o custo mais baixo, C2 e C3, acumula paletes durante o período de recirculação, o seu custo permanece sempre inferior a C4, resultando assim numa utilização sistemática das C2 e C3.

Um efeito semelhante é observado para C2, onde a circulação de paletes neste apenas se dá após a primeira circulação, isto acontece porque no processo do PA a primeira habilidade a ser pedida é sempre a A, logo o custo de atingir A é menor acedendo diretamente, sem requerer uma recirculação, sendo o produto enviado de imediato para a C3 ou C4. Neste contexto a C2 nunca é utilizada como passagem, apenas sendo utilizada após a execução das primeiras duas habilidades.

A passadeira C5 é um caminho obrigatório, logo o seu comportamento apenas reflete o efeito de recirculação e alguma acumulação verificada à entrada no encaminhador, imediatamente antes às passadeiras C6 e C7.

A próxima análise recairá, sobre a comparação entre C6 e C7. Neste caso C6 é ligeiramente mais utilizado numa fase inicial, isto deve-se a alguns PA's escolherem C3 para a execução das duas primeiras habilidades, A e B, e requererem posteriormente a habilidade C, utilizando-se então C6 como passagem, devido à necessidade de recircular a palete. Quando o sistema entra numa fase de um grande número de recirculações, estas passadeiras passam a ter um comportamento semelhante. Embora C7 seja muito mais curta a métrica escolhida penaliza-o no caso de ter alguma carga. Com esta situação a carga é partilhada igualmente pelas duas.

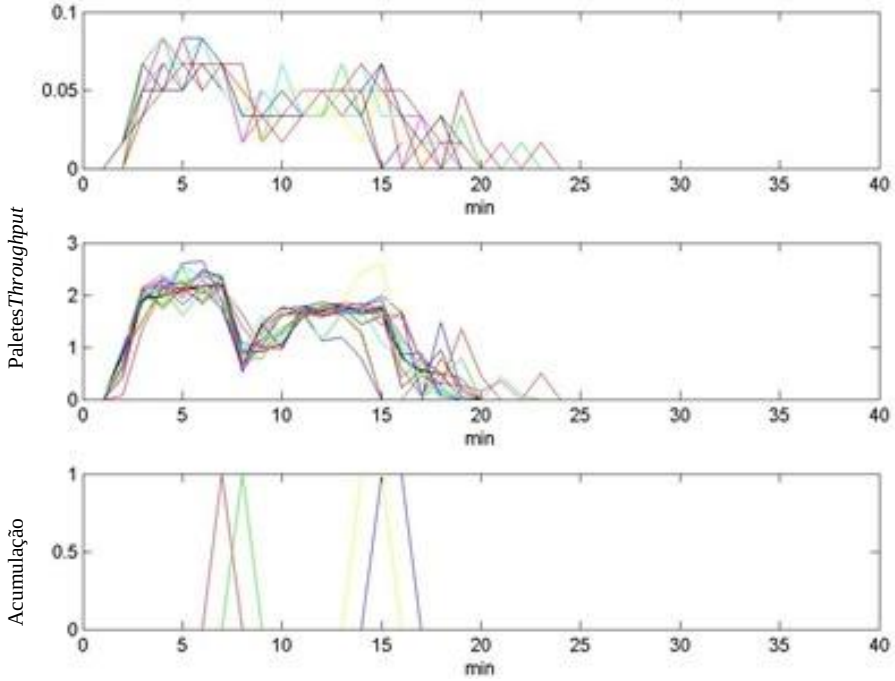


Figura 47 - Comportamento da passadeira C6

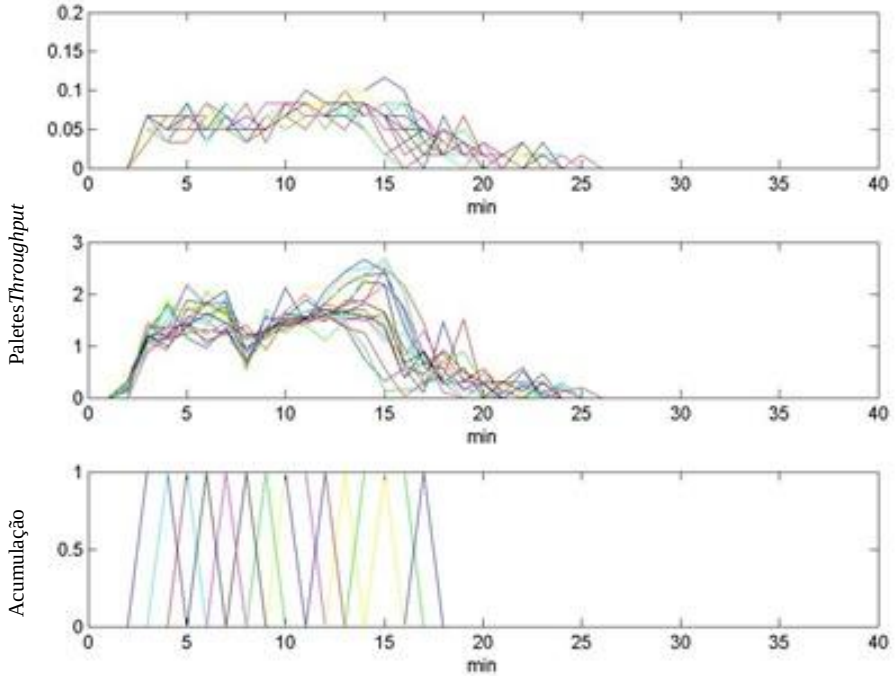


Figura 48 - Comportamento da passadeira C7

Na passadeira C8 evidencia-se um comportamento semelhante ao da C5. Não há, de resto, muito mais a acrescentar na análise a esta passadeira.

O conjunto de passadeiras que serão analisadas de seguida é de extrema importância, sendo elas a C9 e C10, conjuntamente com a C13.

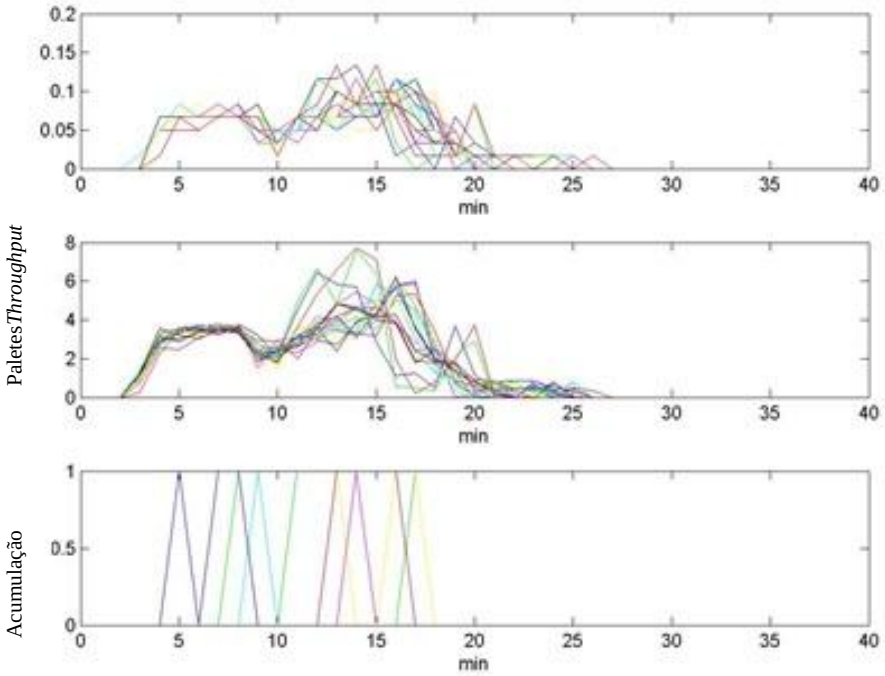


Figura 49 - Comportamento da passadeira C9

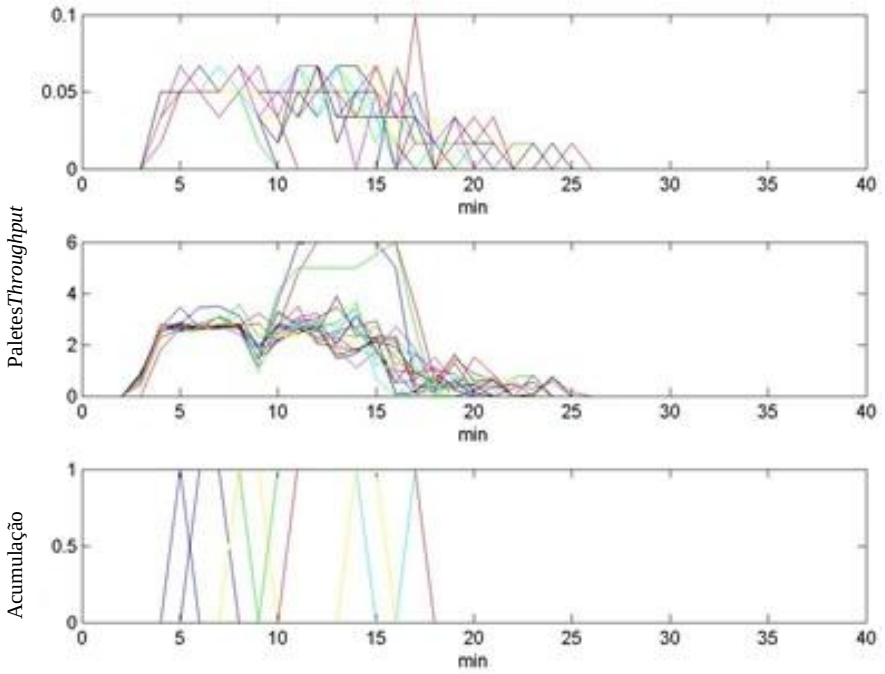


Figura 50 - Comportamento da passadeira C10

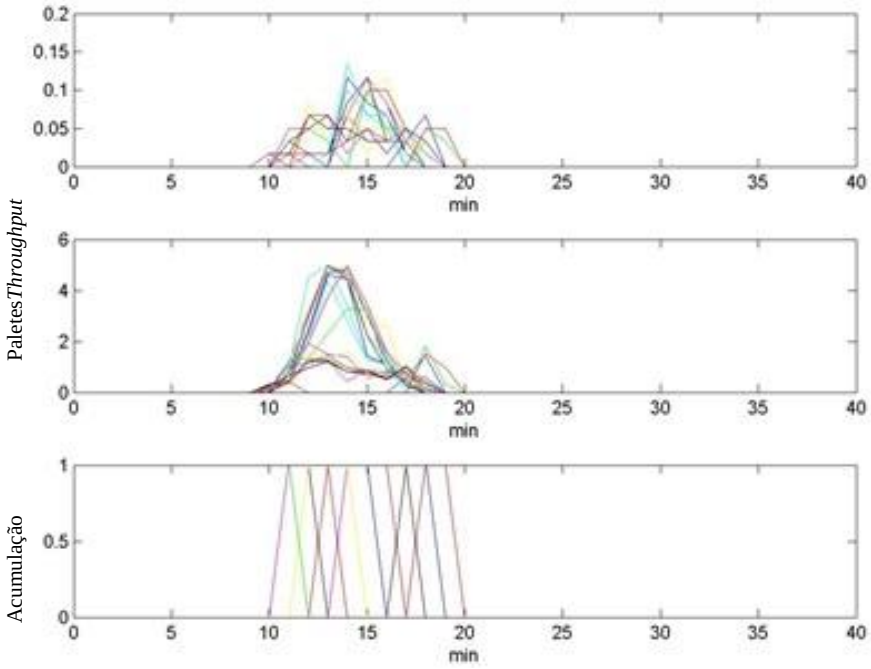


Figura 51 - Comportamento da passadeira C13

Conforme já foi detalhado anteriormente, as passadeiras C9 e C10 constituem uma das primeiras armadilhas para o algoritmo utilizado. No entanto o seu comportamento é bastante semelhante. Esta semelhança no comportamento pode ser explicada como sendo, uma compensação utilizada pelo algoritmo, de forma a compensar a sua miopia, através de uma recirculação rápida das paletes pela passadeira C13, tornando-se assim uma alternativa para todos os PA's do tipo 1 e 2 que executem a habilidade D na passadeira C10. Esta situação acaba por tornar clara a crescente utilização das passadeiras C6 e C7, aquando da aproximação ao minuto dez. Também é interessante analisar alguns dos testes, como se pode ver na Figura 50, que advêm do pior caso, caso esse que ocorre quando há um grande numero de PA's a executar a habilidade D no C10 e posteriormente executam a E também na passadeira C10, obrigando a uma recirculação.

Quanto à passadeira C11 segue um comportamento semelhante à C5 e C8, deste modo nada mais há a acrescentar.

Na próxima análise, a atenção recairá sobre as passadeiras C12, C14 e C15. Nesta parte do sistema, e à semelhança do que acontece nas C9 e C10, existem estações que disponibilizam as mesmas habilidades, em ramos paralelos. C12 (Figura 52) e C15 (Figura 53) denotam um comportamento bastante semelhante, com a exceção do pico inicial, em C15, e o pico médio em C12.

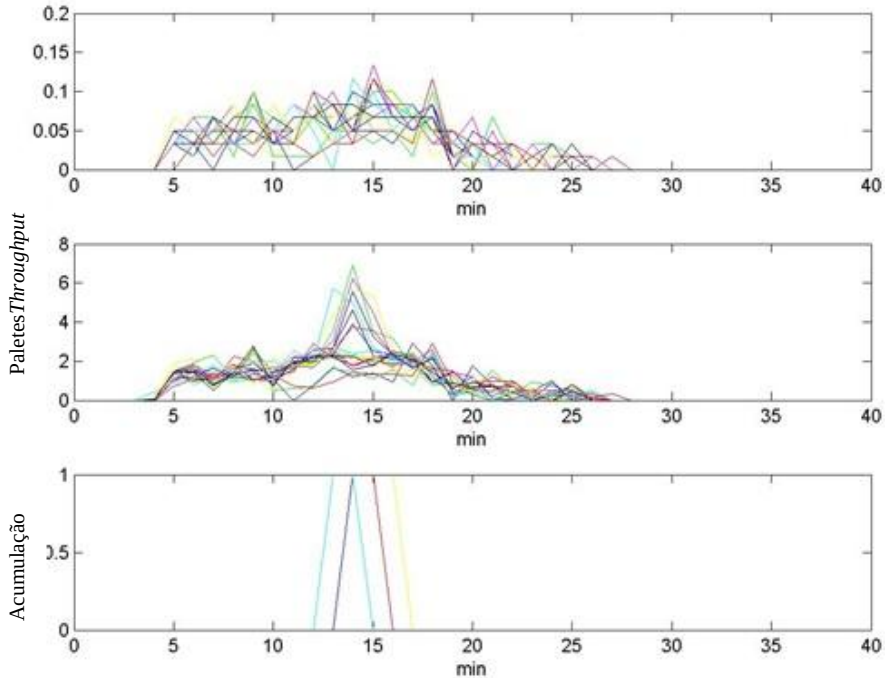


Figura 52 - Comportamento da passadeira C12

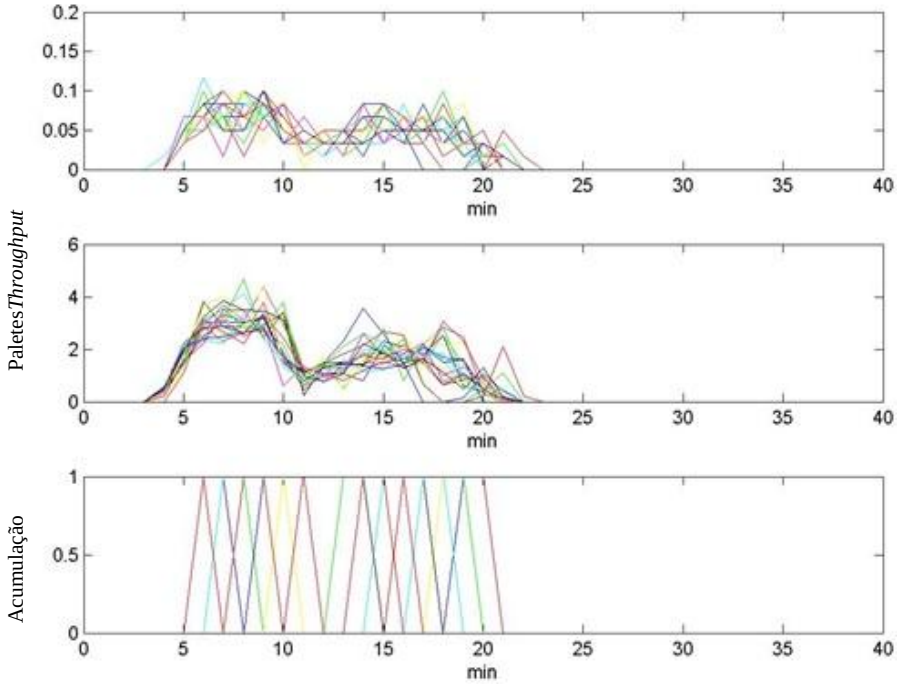


Figura 53 - Comportamento da passadeira C15

C15 tende a ser a via preferencial para a primeira recirculação, onde todos os PA's foram submetidos à execução das habilidades A e B, e agora exigem a execução da habilidade C. Como o desempenho de C15 de C15 se degrada, com o aumento de paletes presente, algumas delas são encaminhadas para a rota alternativa descrita pela C12 e C14 (Figura 54).

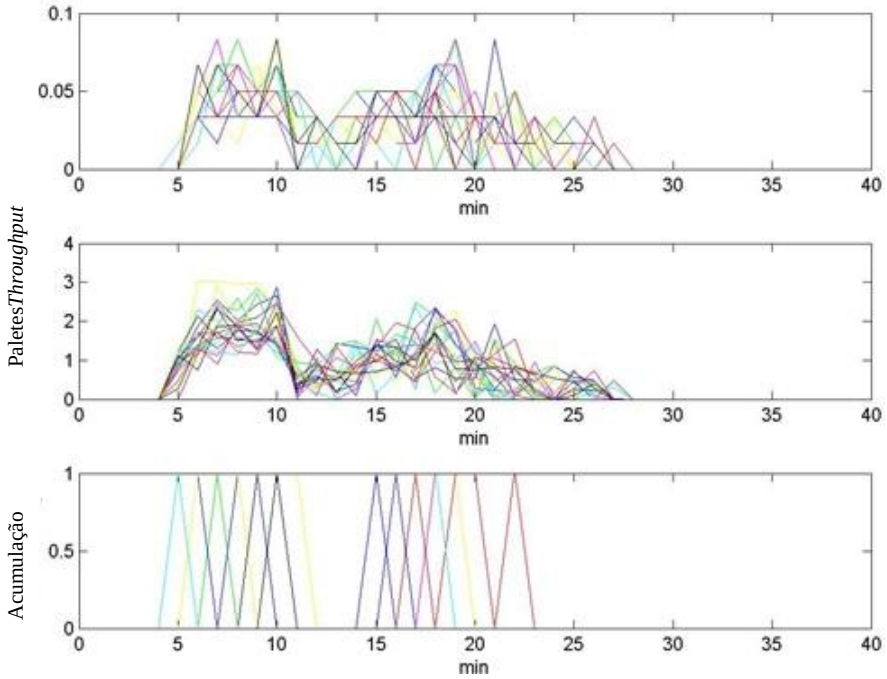


Figura 54 - Comportamento da passadeira C14

O primeiro pico de atividade é em C14, e este deve-se, principalmente, ao efeito de recirculação. A execução da habilidade H, para o produto do tipo 1, acontece sempre depois do primeiro pico de carga. Após este efeito de recirculação há um novo pico menos intenso em torno dos dezassete minutos, que corresponde, a PA's do tipo um que foram recirculados para C12 (Figura 52) e C13 (Figura 51), de forma a poderem mais tarde serem encaminhados para C14, de forma a terminar o seu plano.

O comportamento de C17 e C16 é o esperado, fazendo um balanceamento da carga entre ambos, numa fase inicial funcionam as duas como passagem, até que todos os PA's começam a convergir para o final do plano de execução.

Quanto às passadeiras C18 e C19, estas seguem o padrão de recirculação já descrito, devido ao facto de não existir qualquer rota alternativa.

Os resultados obtidos sugerem que realmente é possível equilibrar a carga de produtos para as estações num ambiente auto-organizado de agentes. É também evidente que a miopia presente no sistema leva a desnecessárias recirculações do produto, de forma a tornar possível a execução do plano. Contudo é importante reter que, não só estes padrões são estáveis, como também o algoritmo de encaminhamento é capaz de, eficazmente encaminhar paletes nestas circunstâncias. É também de elevado interesse, a possibilidade do comportamento de todo o sistema poder ser afinado com base no custo retirado da métrica.

6. Conclusões e Trabalho Futuro

6.1. Conclusões

Com o trabalho desenvolvido ficou assim comprovado que uma arquitetura multiagente auto-organizada pode ser utilizada no suporte a um sistema transporte num ambiente de produção.

Os resultados obtidos neste trabalho sugerem que uma resposta auto-organizada consistente e inserida num ambiente composto por agentes pode ser utilizada com a finalidade de equilibrar a carga e melhorar o desempenho num sistema de produção. A solução proposta desafia as abordagens tradicionais, que habitualmente trabalham esta problemática da vertente do escalonamento.

A abordagem proposta poderá resultar numa alternativa válida, em sistemas onde a dinâmica dos sistemas torna os paradigmas tradicionais inaplicáveis. Devido a estas caraterísticas, estes sistemas são normalmente considerados imprevisíveis. Os resultados obtidos sugerem, pelos padrões de encaminhamento, que um sistema auto-organizado neste caso tem um comportamento que, embora não seja previsível, é pelo menos expectável.

Estas conclusões retiradas pela análise aos testes realizados em ambiente simulado, onde inúmeras perturbações foram provocadas, sobre o sistema de transporte. Nestes casos o sistema de transporte teve uma resposta consistente durante todos os ensaios verificados.

Os testes realizados em ambiente real sobre a forma de protótipo revelaram-se importantes indicadores sobre a aplicabilidade da arquitetura em sistemasreais de produção. Contudo a utilização desta num ambiente real de produção requer um melhoramento no desempenho do sistema. As melhorias deveram passar pela maior rapidez de troca de informação entre elementos do sistema e maior rapidez no processamento da mesma. O desempenho é ponto-chave quando se fala neste tipo de sistemas devido a estes serem sistemas de tempo real.

6.2. Trabalho Futuro

Como trabalho futuro é necessário continuar a estudar a auto-organização em ambientes mecatrónicos, de forma a perceber melhor quais as suas mais-valias e limitações em ambientes com estas condições.

De uma forma específica é necessário um estudo sobre qual o impacto real em sistemas deste género do atraso verificado na troca de informação entre as entidades. Neste caso a troca de mensagens influência o desempenho do mesmo mas é necessária uma compreensão muito mais pormenorizada sobre o impacto verificado.

Outro ponto essencial de revisão é o impacto que a miopia do sistema tem sobre o desempenho do sistema no encaminhamento dos produtos. Após a compreensão do real impacto desta limitação sobre o sistema é necessário trabalhar numa resolução que permita dar a volta a esta caraterística da arquitetura proposta.

Como trabalho inicial prevê-se que este se debruce sobre a compreensão da auto-organização destes sistemas e continuar o trabalho já anteriormente iniciado da desmistificação de algumas caraterísticas frequentemente atribuídas a sistemas destas naturezas como imprevisibilidade e aleatoriedade.

Durante o trabalho realizado foi escrito um artigo, (Ribeiro, Rocha et al. 2012). Este foi apresentado na conferência IECON (http://www.iecon2012.org).

Também no âmbito deste trabalho está a ser desenvolvido um artigo para posterior submissão numa revista cientifica internacional.

7.Referências

Barata, J. (2005). Uma Aproximação Baseada em Coligações para a Agilidade da Planta Fabril Usando Multiagentes, FCT/UNL.

Bellifemine, F., A. Poggi, et al. (1999). JADE–A FIPA-compliant agent framework. Proceedings of PAAM, London.

Blum, C. (2005). "Ant colony optimization: Introduction and recent trends." Physics of Life Reviews **2**(4): 353-373.

Boysen, N., M. Fliedner, et al. (2008). "Assembly line balancing: Which model to use when?" International Journal of Production Economics **111**(2): 509-528.

Erol, R., C. Sahin, et al. (2012). "A multi-agent based approach to dynamic scheduling of machines and automated guided vehicles in manufacturing systems." Applied Soft Computing **12**(6): 1720-1732.

Galan, R., J. Racero, et al. (2007). "A systematic approach for product families formation in Reconfigurable Manufacturing Systems." Robotics and Computer-Integrated Manufacturing **23**(5): 489-502.

Giordani, S., M. Lujak, et al. (2013). "A distributed multi-agent production planning and scheduling framework for mobile robots." Computers & Industrial Engineering **64**(1): 19-30.

Gou, L., P. B. Luh, et al. (1998). "Holonic manufacturing scheduling: architecture, cooperation mechanism, and implementation." Computers in Industry **37**(3): 213-231.

Guo, Q.-l. and M. Zhang (2010). "An agent-oriented approach to resolve scheduling optimization in intelligent manufacturing." Robotics and Computer-Integrated Manufacturing **26**(1): 39-45.

Haneyah, S. W. A., J. M. J. Schutten, et al. (2013). "Generic planning and control of automated material handling systems: Practical requirements versus existing theory." Computers in Industry **64**(3): 177-190.

Hsieh, C.-H., C. Cho, et al. (2012). "Simulation study for a proposed segmented automated material handling system design for 300-mm semiconductor fabs." Simulation Modelling Practice and Theory **29**(0): 18-31.

Jahromi, M. H. M. A. and R. Tavakkoli-Moghaddam (2012). "A novel 0-1 linear integer programming model for dynamic machine-tool selection and operation allocation in a flexible manufacturing system." Journal of Manufacturing Systems **31**(2): 224-231.

Kanaga, E. G. M. and M. L. Valarmathi (2012). "Multi-agent based Patient Scheduling Using Particle Swarm Optimization." Procedia Engineering **30**(0): 386-393.

Kennedy, J. and R. Eberhart (1995). Particle swarm optimization. Neural Networks, 1995. Proceedings., IEEE International Conference on.

Koren, Y., U. Heisel, et al. (1999). "Reconfigurable Manufacturing Systems." CIRP Annals - Manufacturing Technology **48**(2): 527-540.

Lee, K., B.-C. Choi, et al. (2009). "Approximation algorithms for multi-agent scheduling to minimize total weighted completion time." Information Processing Letters **109**(16): 913-917.

Leitão, P. and F. Restivo (2006). "ADACOR: A holonic architecture for agile and adaptive manufacturing control." Computers in Industry **57**(2): 121-130.

Leung, C. W., T. N. Wong, et al. (2010). "Integrated process planning and scheduling by an agent-based ant colony optimization." Computers & Industrial Engineering **59**(1): 166-180.

Li, J., X. Dai, et al. (2009). "Rapid design and reconfiguration of Petri net models for reconfigurable manufacturing cells with improved net rewriting systems and activity diagrams." Computers & Industrial Engineering **57**(4): 1431-1451.

Li, X., C. Zhang, et al. (2010). "An agent-based approach for integrated process planning and scheduling." Expert Systems with Applications **37**(2): 1256-1264.

Onori, M., H. Alsterman, et al. (2005). An architecture development approach for evolvable assembly systems. Assembly and Task Planning: From Nano to Macro Assembly and Manufacturing, 2005. (ISATP 2005). The 6th IEEE International Symposium on.

Ribeiro, L. and J. Barata (2011). "Re-thinking diagnosis for future automation systems: An analysis of current diagnostic practices and their applicability in emerging IT based production paradigms." Computers in Industry **62**(7): 639-659.

Ribeiro, L., J. Barata, et al. (2010). Evolvable Production Systems: An Integrated View on Recent Developments. Proceedings of the 6th CIRP-Sponsored International Conference on Digital Enterprise Technology. G. Huang, K. L. Mak and P. Maropoulos, Springer Berlin Heidelberg. **66:** 841-854.

Ribeiro, L., A. Rocha, et al. (2012). A product handling technical architecture for multiagent-based mechatronic systems. IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society, IEEE.

Sabar, M., B. Montreuil, et al. (2009). "A multi-agent-based approach for personnel scheduling in assembly centers." Engineering Applications of Artificial Intelligence **22**(7): 1080-1088.

Sallez, Y., T. Berger, et al. (2009). "A stigmergic approach for dynamic routing of active products in FMS." Computers in Industry **60**(3): 204-216.

Savkin, A. V. and J. Somlo (2009). "Optimal distributed real-time scheduling of flexible manufacturing networks modeled as hybrid dynamical systems." Robotics and Computer-Integrated Manufacturing **25**(3): 597-609.

Tanenbaum, A. S. (1996). Computer networks, Prentice Hall Englewood Cliffs.

Ueda, K. (1992). A Concept for Bionic Manufacturing Systems Based on DNA-type Information. Proceedings of the IFIP TC5 / WG5.3 Eight International PROLAMAT Conference on Human Aspects in Computer Integrated Manufacturing, North-Holland Publishing Co.**:** 853-863.

Van Brussel, H., J. Wyns, et al. (1998). "Reference architecture for holonic manufacturing systems: PROSA." Computers in Industry **37**(3): 255-274.

Vrba, P., Tichy, et al. (2011). "Rockwell Automation's Holonic and Multiagent Control Systems Compendium." Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on **41**(1): 14-30.

Wong, M. M., C. H. Tan, et al. (2007). "On-line reconfiguration to enhance the routing flexibility of complex automated material handling operations." Robotics and Computer-Integrated Manufacturing **23**(3): 294-304.

Xiang, W. and H. P. Lee (2008). "Ant colony intelligence in multi-agent dynamic manufacturing scheduling." Engineering Applications of Artificial Intelligence **21**(1): 73-85.

Zhenqiang, B., W. Weiye, et al. (2012). "Research on Production Scheduling System with Bottleneck Based on Multi-agent." Physics Procedia **24**: 1903-1909.