Faculdade de Engenharia da Universidade do Porto

Simulador de Sistemas de Produção e de Informação Industriais

Aplicação a sistema de produção *lean*

Ricardo Arnaldo da Cunha Fernandes

Relatório de Projecto realizado no âmbito do
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major Automação

Orientador: Prof. Dr. José Faria

Junho de 2008

Resumo

O projecto aqui apresentado teve como principal objectivo conhecer a ferramenta de simulação AnyLogic e avaliá-la na perspectiva da simulação de sistemas de produção, em especial de sistemas de produção *lean*. Pretendia-se também avaliar o interesse e as vantagens da utilização de ferramentas de simulação no apoio a projectos *lean*.

A ferramenta Anylogic reúne num só ambiente diferentes paradigmas de simulação, sistemas a eventos discretos, dinâmica de sistemas e sistemas baseados em agentes, o que, aliado ao facto de se tratar de uma ferramenta recente, baseada na linguagem Java e em modelação orientada a objectos, constituiu um factor de motivação.

Na primeira fase do projecto, foi efectuado um estudo aprofundado da ferramenta que incidiu sobre os recursos de modelação disponíveis e foram desenvolvidos vários cenários de simulação com o objectivo de explorar as diferentes funcionalidades oferecidas pela ferramenta.

Na segunda fase do projecto, foi elaborado o modelo de simulação de um sistema de produção constituído por quatro células de montagem, uma célula de preparação de componentes, quatro *milk runs* e um supermercado de componentes.

Começou por ser efectuada uma modelação da estrutura e da dinâmica do sistema de produção recorrendo a modelos UML – diagramas de classes, diagramas de sequência e diagramas de estados. O modelo de simulação em AnyLogic foi então criado a partir destes modelos. Foi também criada a animação gráfica do modelo que permite visualizar, em tempo real, a movimentação das entidades e os indicadores de desempenho relevantes.

O paradigma de simulação baseado em agentes revelou-se particularmente bem adaptado à simulação de sistemas de produção *lean*. De facto, cada um dos elementos destes sistemas (por exemplo células e *milk runs*) possui um elevado grau de autonomia, o que se ajusta com naturalidade à filosofia de simulação baseada em agentes. De facto, foi possível estabelecer um mapeamento quase directo entre as entidades do sistema de produção e os objectos de modelação proporcionados pelo AnyLogic.

A partir do modelo de simulação, foram criados vários cenários relativos ao nível dos buffers do supermercado e ao número de trabalhadores da célula de preparações e foram definidos os indicadores que permitem avaliar a performance do sistema.

Os resultados obtidos através do modelo de simulação permitem comparar diferentes soluções relativamente ao funcionamento do sistema.

Abstract

The project presented had as primarily objective to know the simulation tool AnyLogic and to evaluate it in simulation of production systems, especially in lean production systems. The aim was to also assess the interest and the advantages of using simulation tools in support of lean projects.

The tool Anylogic combines in a single environment different paradigms of simulation, the discrete events systems, systems dynamic's and systems based on agents, and coupled with the fact that this is a new tool, based on Java and object oriented, was a source of motivation.

In the first phase of the project, was made a detailed study of the tool which was focused on the available resources and have been developed various scenarios of simulation with the aim of exploring the multiple features offered by the tool.

In the second phase of the project, have been developed the simulation of a production system constituted by four cells of assembly, a cell of components preparation, four milk runs and a supermarket of components.

The model of the structure and the dynamics of the production system were made by using UML models - class diagrams, state charts and sequence diagrams. The simulation model in AnyLogic was then created from these models. It was also created the model's animation for displaying in real time, the entities handling and the relevant performance indicators.

The simulation paradigm of systems based on agents proved to fit well on the simulation of lean production systems. Indeed, each of the elements of these systems (such cells and milk runs) has a high degree of autonomy, which fits naturally with the philosophy of simulation of systems based on agents. Also, it was almost possible to establish a direct mapping between the entities of the production system and the modeling objects provided by AnyLogic.

From the simulation model were set up various scenarios concerning the level of the supermarket buffers and the number of employees of the preparations cell, and were defined indicators which measure the performance of the system.

The results obtained through the simulation model made possible to compare different solutions for the system functioning.

Agradecimentos

Agradeço ao meu orientador, o Prof. Dr. José Faria a sua orientação e coordenação durante a realização dos trabalhos conducentes à elaboração deste projecto, assim como todo o trabalho que teve durante a sua orientação. Agradeço ainda o esforço desenvolvido na leitura e as sugestões de revisão que permitiram o enriquecimento do texto deste projecto.

Agradeço aos meus pais e irmão, o todo o incentivo e apoio que sempre me deram.

Índice

Lista de figuras

Lista de tabelas

Abreviaturas

Lista de abreviaturas

ECSL	*Extended Control and Simulation Language*
FEUP	Faculdade de Engenharia da Universidade do Porto
FIFO	*First In First Out*
GPSS	*General Purpose Simulation System*
SIMAN	*SIMulations ANalysis*
UML	*Unified Modeling Language*

Capítulo 1

Introdução

1.1 - Objectivos

A realidade actual está marcada pela competição das empresas, em mercados caracterizados pela procura de produtos de qualidade elevada e com custos de produção e prazos de entrega os mais baixos possíveis. Para que seja possível às empresas manterem-se competitivas é necessário ajustar e melhorar permanentemente os seus sistemas de produção.

A simulação computacional é uma das ferramentas que as empresas podem utilizar para adquirir e organizar o conhecimento necessário para adaptarem os seus sistemas de produção às novas exigências do mercado

O projecto que está na origem desta tese teve como principal objectivo conhecer a ferramenta de simulação AnyLogic e avaliá-la na perspectiva da simulação de sistemas de produção, em especial sistema de produção *lean*. Em particular, pretendia-se avaliar o interesse em utilizar ferramentas de simulação no apoio a projectos *lean*.

O AnyLogic é uma ferramenta relativamente recente, tendo a sua versão actual sido lançada em Abril de 2007. Trata-se de uma ferramenta multi-paradigma, pois permite a combinação dos três grandes paradigmas de simulação, sistemas a eventos discretos, com a dinâmica de sistemas e sistemas baseados em agentes, tudo num único ambiente de desenvolvimento integrado. Esta capacidade torna-a uma ferramenta bastante poderosa que permite a modelação de sistemas complexos com elevado nível de detalhe e que, como se verá, é particularmente bem adaptada à simulação de sistemas de produção *lean*.

Para avaliar a ferramenta, foi desenvolvido um caso de estudo baseado no sistema de produção de uma grande empresa industrial.

Este projecto decorreu em estreita articulação com os projectos *lean* realizados por outros colegas em ambiente industrial.

O caso de estudo para a avaliação da ferramenta foi baseado num desses projectos, o qual tinha por objectivo a criação de um sistema *pull* numa célula de produção.

Além de permitir avaliar a ferramenta, este projecto também constituiu um primeiro passo para a criação de ferramentas de simulação para apoio a projectos *lean*.

1.2 - Estrutura do documento

O presente projecto encontra-se estruturado em seis capítulos, cujo conteúdo é sumariamente descrito de seguida.

No primeiro capítulo, é realizado o enquadramento do trabalho, são definidos os seus objectivos, é delineada a sua estrutura, bem como apresentada a metodologia seguida na sua elaboração.

No segundo capítulo, com o título *Fundamentos de Simulação*, é apresentada uma perspectiva geral da simulação, abordando temas e conceitos básicos de simulação, tais como: vantagens e desvantagens, metodologia de simulação, tipos de aplicação e ferramentas de simulação.

No capítulo terceiro, denominado *Apresentação do software AnyLogic*, é feita uma apresentação da ferramenta de simulação AnyLogic. Esta apresentação conta com temas como: a modelação de sistemas a eventos discretos, modelação da dinâmica de sistemas, modelação de sistemas baseados em agentes, a AnyLogic *Enterprise Library*, Agente, Interacção entre objectos.

No quarto capítulo intitulado, *Apresentação e modelação do sistema em estudo,* é apresentada uma descrição global do sistema de produção onde se insere a célula que será objecto do caso de estudo. Aqui, além da descrição do sistema físico, é apresentado a respectiva modelação, tanto na componente estrutura como na componente dinâmica. São ainda definidas as entidades que constituem o sistema, as relações entre essas entidades e os indicadores de desempenho que serão utilizados para avaliar as várias soluções em confronto.

No capítulo quinto, designado *Desenvolvimento do modelo de simulação*, é mostrado como as entidades definidas no modelo do sistema foram transpostas para o modelo de simulação implementado no AnyLogic. Neste capítulo também é apresentada a animação do sistema desenvolvida e os diferentes cenários em avaliação e os respectivos resultados

No sexto e último capítulo, cujo título é *Conclusões e trabalho futuro*, é feita a uma avaliação geral do trabalho, são referidas as principais dificuldades encontradas na sua realização. São também apontadas linhas de desenvolvimento possíveis que dêem continuidade e aprofundem os resultados já alcançados no âmbito deste projecto.

Capítulo 2

Fundamentos de Simulação

2.1 - Introdução

Nos dias de hoje e devido à globalização instalada, as empresas necessitam de ser altamente flexíveis, para assim conseguirem adaptar-se às necessidades do mercado e consequentemente ter uma maior e mais rápida capacidade de resposta, a essas mesmas exigências. Assim sendo, surgem processos cada vez mais complexos, os quais necessitam de análise e avaliação de desempenho.

Devido ao elevado grau de complexidade dos processos, na maioria dos casos é impossível, ou por outro lado os custos são elevados, realizar um sistema físico real para o caso em estudo.

Contudo existem métodos que possibilitam a representação do sistema. Métodos tais como a simulação.

A simulação é uma das mais poderosas ferramentas de análise de desempenho de um sistema ou processo, através da formulação de um modelo matemático, o qual deve reproduzir, do modo mais fiel possível, as características do sistema original. Manipulando o modelo e analisando os resultados, pode-se concluir como os diversos factores irão afectar o desempenho do sistema (1).

Simular envolve a modelação de um processo ou sistema de tal modo que o modelo imita a resposta, do sistema actual, a um evento que ocorre *a posteriori* (2).

A Simulação torna possível o estudo, a análise e a avaliação de diversas situações que não poderiam ser conhecidas de outro modo. Num mundo cada vez mais competitivo, a simulação tornou-se numa metodologia de resolução de problemas indispensável quer para engenheiros, quer para gestores de topo (3). Por esta razão, o número de empresas ou organizações que hoje se socorrem da simulação como método para optimizar o seu desempenho, tem vindo a aumentar (4).

Simulação é nada mais, nada menos, que a técnica de fazer experiências amostrais no modelo de um sistema. As experiências são feitas no modelo, ao invés de no próprio sistema real, porque é mais conveniente e menos dispendioso (5).

A função primária de um modelo de simulação é examinar como o sistema se comporta durante um período de tempo. Para atingir este objectivo, o modelo deve providenciar

facilidades, para representar o estado actual do sistema, e várias pré-condições que se satisfeitas, irão resultar num estado futuro. (6)

Simular é desenvolver um modelo de um sistema em projecto ou já construído com o objectivo de avaliar o seu comportamento segundo vários aspectos, sendo possível desta forma tirar conclusões sem necessitar de o construir (7).

A simulação pode ser vista como uma metodologia que procura (3):

- Descrever o comportamento dos sistemas.
- Construir teorias e hipóteses que demonstram o comportamento observado.
- Usar o modelo para prever o futuro, isto é, os efeitos que serão produzidos pelas alterações no sistema ou no seu método de operação.

2.2 - Vantagens e desvantagens

De seguida são mostradas vantagens e desvantagens referenciadas por vários autores (8) (9) (3).

Entre as principais vantagens da simulação contam-se as seguintes:

- Permite uma melhor a compreensão dos sistemas pelo desenvolvimento de um modelo do sistema em questão, e pela observação do sistema em operação por longos períodos de tempo.
- Testar hipóteses de viabilidade do sistema.
- Comprimir o tempo para observar um determinado fenómeno por longos períodos de tempo ou expandir o tempo para observar um fenómeno complexo em detalhe.
- Estudar os efeitos de determinadas mudanças na operação do sistema pela alteração do modelo, isto não pode ser feito sem “corromper” o sistema real o que reduz o risco de experimentar no sistema real.
- Experimentar novas ou desconhecidas situações para as quais a informação disponível é fraca.
- Permite identificar as variáveis mais importantes na performance e como interagem.
- Identificar estrangulamentos no fluxo de entidades (materiais, pessoas, etc) ou informação.
- O uso de múltiplas medições para analisar as configurações dos sistemas.
- Desenvolver sistemas mais robustos com melhor arquitectura num tempo de desenvolvimento mais curto.
- A informação recolhida por simulação é normalmente mais barata do que a informação recolhida usando o sistema real.
- Pode ser usada para explorar novas políticas de escalonamento dos recursos, procedimentos operativos, regras de decisão, estruturas organizacionais, fluxos de informação, sem ser necessário interromper o normal funcionamento do sistema.
- Permite testar novas arquitecturas de *hardware*, *layouts* físicos, *software*, sistemas de transporte etc. Antes de se comprometer com os recursos para a sua implementação.
- Permite testar hipóteses explicativas de como ou porquê determinado fenómeno ocorre no sistema.

Por outro lado, as principais desvantagens apontadas à simulação são as seguintes:

- Em algumas situações as animações visuais combinadas com a pressão temporal presente em todos os projectos, pode levar a decisões prematuras baseadas em evidências insuficientes.
- A construção de modelos é uma arte que requer treino especializado. A qualidade da análise depende directamente da qualidade do modelo e da qualidade do modelador.
- Os resultados da simulação são algumas vezes difíceis de interpretar.
- Um modelo de um sistema complexo pode ter um custo elevado e levar vários meses para ser desenvolvido, especialmente nos casos em que os dados são de difícil obtenção e não coerentes.

2.3 - Metodologia de simulação

No caso geral e segundo Carson (10) a simulação envolve as seguintes etapas, embora nem todas as etapas tenham sido abordadas neste estudo.

Formulação do problema – Todas as actividades de modelização devem ser focadas no objectivo.

Por vezes, o problema não é conhecido ou então pouco compreendido, e a formulação do problema é iniciada em termos de observação dos sintomas. Com o avançar do estudo, a natureza do problema vai ficando mais clara, e a formulação do problema pode ser reiniciada e clarificada com a equipa de projecto.

Durante esta fase, a equipa de projecto deve desenvolver uma lista de questões específicas às quais o modelo deve responder e desenvolver uma lista de indicadores de performance que irão avaliar e comparar as alternativas a ser modeladas. Por vezes, o cliente tem um determinado objectivo em mente, por exemplo, que o novo sistema com um determinado nível de recursos terá uma determinada resposta. Isto significa que se o estudo verificar que o desenho do sistema proposto ou o conjunto de regras de operação não alcança o resultado expectável, então é esperado que o modelo forneça informação e realce as causas para tal acontecimento, para que assim a equipa desenvolva alternativas inteligentes que tenham melhores hipóteses de alcançar os objectivos esperados.

Nesta fase, o líder da equipa de projecto necessita de colocar questões a todos os participantes e desenvolver um conjunto de pressupostos que irão ser a base do desenvolvimento do modelo. Três importantes considerações gerais são:

- Contexto e limites do modelo.
- Nível de detalhe.
- Foco do projecto.

O contexto e limites do modelo determinam o que está e o que não esta no modelo. O nível de detalhe, especifica qual a profundidade que um componente ou entidade é modelado, isto é determinado pelas questões levantadas e pela disponibilidade dos dados. Imagine-se os limites como uma largura e o nível de detalhe com uma profundidade. O foco do projecto é as questões, para as quais o modelo será usado para responder.

Em suma, a formulação do problema deve desenvolver as seguintes actividades:

- Reunir todas as partes interessadas no arranque do projecto, para executar a formulação inicial do problema e discutir os pressupostos do modelo. Se estiver alguém a representar o cliente nas reuniões de revisão ou na apresentação final, essa pessoa deve assistir às reuniões iniciais. Se esta pessoa entende que o modelo deve responder a determinadas questões, então essas questões devem ser colocadas no arranque do projecto.
- Documentar todos os pressupostos e todos o requisitos de dados. Incluir objectivos, questões específicas que necessitam de resposta e indicadores de performance.

Construção do modelo – é constituída por duas grandes fases:

- Desenvolvimento da estrutura de dados que representa os dados necessários do modelo.
- A tradução dos requisitos para a linguagem ou representação requerida pelo software de simulação.

O analista de simulação deve construir uma estrutura de dados que represente os dados e as suas interligações, e de seguida, encaixar esta estrutura no software de simulação.

Definição de dados de entrada - Normalmente o cliente já tem os dados de entrada. Os dados podem incluir base de dados, livros de registos, estudos de amostragem e estudos temporais. Infelizmente nem sempre, e são poucas as vezes em que todos os dados estão disponíveis e com a qualidade necessária. Nestes casos é fundamental fazer um esforço para recolher os dados em falta. A apresentação dos dados deve ter uma formatação adequada à formatação a ser utilizada posteriormente no modelo.

Verificação - Na verificação do modelo, o analista de simulação verifica o modelo, usando de diferentes técnicas, para verificar que o modelo funciona de acordo com os pressupostos acordados. Isto é mais do que corrigir erros na programação. Todas as saídas do modelo devem fazer sentido e ser razoáveis para um conjunto de parâmetros de entrada. Várias técnicas devem ser aplicadas, incluindo, mas não só:

- Ensaios com uma vasta gama de parâmetros e números aleatórios.
- Uma profunda revisão de todas as saídas do modelo, e não apenas dos indicadores de performance primários, mas de vários indicadores secundários.
- Utilização do *debuger* do software.
- Revisão de profissionais com mais experiência em simulação.

Uma atitude relevante a tomar é a do método científico. Em primeiro lugar, formular uma hipótese: o modelo está correcto. Em segundo lugar, fazer o máximo esforço para provar que a hipótese é falsa, ou seja, tentar provar que o modelo não é bom em algum aspecto. Se após um grande esforço, não existirem provas de um modelo defeituoso, então concluir que o modelo está verificado. Do ponto de vista científico, o melhor que pode ser alcançado é uma tentativa de verificação. Um futuro teste, ou uma mudança de condições ou de dados, pode detectar um problema no modelo que exija mudanças. Em termos simples existe quase um número ilimitado de potenciais testes que podem ser realizados para testar a validade de um modelo. Na prática, o tempo apenas permite realizar um determinado número. Então, o melhor que se pode alcançar é um fracasso para rejeitar a hipótese de um modelo válido.

Validação - Nesta fase existe o envolvimento do cliente. Após o analista estar convencido de que o modelo está exacto e verificado, deve realizar uma profunda revisão do modelo com a equipa do cliente. É importante ter todos os membros da equipe do cliente que possam ter interesse no modelo, e os quais esperam que o modelo tenha capacidade responder a todas suas perguntas. Várias técnicas, semelhantes às utilizadas durante a verificação, podem ser utilizadas durante a validação modelo, incluindo a utilização de animações e de outros elementos visuais para apresentar o modelo. Se foram recolhidos dados suficientes do sistema real que correspondem a uma das configurações possíveis do modelo, mais testes podem ser realizados comparando o sistema real com o modelo.

Experimentação - O plano desenvolvido durante o arranque projecto fornece as primeiras orientações para uma série de experiências. Normalmente, a simulação é usada para comparar um grande número de alternativas, e talvez para avaliar como maior pormenor um pequeno número de alternativas recomendadas. Nos pressupostos deve constar uma descrição das variações esperadas para o modelo, incluindo a gama de variação de cada parâmetro de entrada a ser simulado, para assim representar as alternativas de interesse.

Na prática, as experiências iniciais levantam novas questões, as quais podem mudar a direcção do estudo, após as experiências iniciais serem executadas e analisadas. Por exemplo, as experiências iniciais podem estabelecer uma nova proposta de desenho ou que um conjunto regras operacionais leva a grandes estrangulamentos ou ainda outros problemas, o que levanta a necessidade de repensar o desenho do sistema.

Análise – A análise é baseada nos indicadores de desempenho. Tipicamente em aplicações da indústria transformadora e de logística, existem medidas de fluxo, a utilização dos recursos, filas e estrangulamentos. Muitas vezes as primeiras experiências produzem resultados que identificam um problema, ou sintomas de um problema, mas não fornecem prontamente as causas do problema ou não fornecem informações suficientes para dar uma visão sobre a natureza do problema. Essa visão é crítica, quando a equipa tem de desenvolver sugestões de desenho ou melhorias operacionais, as quais têm hipóteses de resolver o problema identificado. Nesta situação, o analista terá de utilizar o modelo de base para formular hipóteses sobre os eventuais problemas identificados. Em vários projectos de simulação, ao longo de muitos anos, Carson viu a necessidade de utilizar um modelo para procurar as causas do problema que não são óbvias à primeira vista e de conceber novos indicadores de desempenho para confirmar hipóteses sobre as causas de falha do sistema. O conhecimento adquirido é inestimável quando se trata de sugerir mudanças para melhorar o desempenho do sistema, a fim de cumprir a meta estipulada.

Documentação - A apresentação dos resultados de uma experiência geralmente inclui uma ou mais apresentações e a escrita de um relatório. Apresentações permitem perguntas e respostas e expansão de explicações. O relatório final deverá incluir os pressupostos iniciais, devidamente revistos, bem como, evidentemente, os principais resultados e recomendações do estudo.

2.4 - Simulação de sistemas de produção

Uma abordagem de modelação de um sistema de produção será determinada, em parte, pelo tipo de sistema necessário para atender as exigências produção. Por exemplo, o sistema

totalmente autónomo será necessariamente modelado de modo diferente de um sistema que exige várias máquinas, operadores, e estações de trabalho. Além disso, o papel das pessoas como uma componente integrante do sistema tem de ser considerado (11).

A concepção de sistemas de produção é um processo moroso que geralmente envolve a determinação dos seguintes parâmetros:

- Processos de fabrico / máquinas: Tipos de processos, tendo em conta as alternativas, exigidos para o fabrico dos produtos.
- Operação em sequência e rotas necessários para fazer a produção: Depois definidos os processos a sua sequência e rotas deverá ser determinado.
- *Layout* físico: Os processos e máquinas devem ser dispostos num determinado espaço.
- Fluxo de material: O movimento de peças de e para o sistema, carga / descarga de peças para máquinas, meios de transporte dos materiais, tais como passadeiras rolantes, robôs, etc.
- Equipamentos de apoio: Peças de substituição, ferramentas, etc.
- Alocação de tarefas a operários: Normalmente, às pessoas são dadas apenas as tarefas que máquinas não podem executar numa determinada sequência operações. A atribuição de tarefas ao operador, deve ser considerada em paralelo, com o desenvolvimento do sistema de produção para maximizar a sua capacidade, bem como valorizar a contribuição humana. Deve ainda ser analisada a atribuição funcional de uma pessoa a uma máquina e efectuar análises para avaliar a tarefa, tais como, produtividade, utilização, segurança e ergonomia.

Entendendo-se por sistema de produção lean um sistema com as seguintes características fundamentais:

- Não existe um sistema centralizado de controlo da execução das ordens de produção nas várias células. Em vez disso, as ordens de produção são enviadas apenas para as células finais.
- As ordens de produção nas células a montante e os fluxos de materiais entre estas células e as células finais são "puxados" (*pull*) pelas necessidades das células finais.
- Existe uma separação entre as actividades de produção e de logística interna, que são executadas por operadores dedicados.
- Os tempos dos ciclos de produção e de abastecimento são normalizados.

2.5 - Tipos de aplicações

Alguns domínios de aplicações habituais para a simulação, tal como sugerido por Shannon, são as seguintes (3):

Sistemas Computacionais: componentes de *hardware*, *software*, redes de computadores, estruturas e gestão de base de dados, processamento de informação, fiabilidade de *hardware* e *software*.

Manufactura: sistemas de manuseamento e armazenamento de materiais, linhas de montagem, recursos automatizados de produção e armazenamento, sistemas de controlo de stocks, estudos de manutenção, *layout* de unidades fabris, projecto de máquinas.

Negócios: análises de produtos, politica de preços, estratégias de marketing, estudos de aquisição de empresas, análise de fluxo de caixa, previsão, alternativas de transporte, planeamento de aumento de trabalho.

Governo: armamentos e tácticas militares, previsão de crescimento populacional, sistemas de saúde, sistemas contra incêndio, polícia, justiça criminal, projectos de estradas, controlo de tráfego, serviços de saneamento.

Ecologia e Meio Ambiente: poluição e purificação de água, controle de desperdícios, poluição do ar, controlo de pragas, previsões climatéricas, análise de terramotos e tempestades, exploração e extracção mineral, sistemas de energia solar.

Sociedade e Comportamento: análises alimento/população, politicas educacionais, estrutura organizacional, análise de sistemas sociais, administração universitária.

Biociências: análise de performance desportiva, ciclos de vida biológicos, estudos biomédicos.

2.6 - Ferramentas e linguagens de simulação

Para a implementação dos modelos construídos em programas computacionais são utilizadas ferramentas de simulação que podem ser classificadas em três grandes grupos (12).

Linguagens de uso geral

Embora estas linguagens permitam a modelação de várias aplicações, é necessário que o programador tenha bons conhecimentos em programação. O que conduz a enorme esforço no seu desenvolvimento, e tornava em muitos casos inviável o uso da simulação.

Como exemplos de linguagens de uso geral citam-se: o FORTRAN, PASCAL, Java, Visual Basic, C e C++ que são utilizadas no desenvolvimento de muitas aplicações informáticas.

Pacotes de simulação de utilização genérica

Depois de desenvolvidas várias simulações com linguagens de carácter geral, reconheceu-se que em muitos casos os sistemas ou subsistemas modelados, eram iguais ou semelhantes. Surgiram então linguagens de propósito específico.

Assim na década de 60 surgem as linguagens específicas, as quais levaram uma evolução bastante significativa.

São linguagens desenvolvidas com o objectivo específico de facilitar e tornar económico o processo de escrita de programas, para a execução de simulações. Como exemplos deste tipo de linguagens, destacam-se:

- GPSS – desenvolvida por Geoffrey Gordon na década de 60, utiliza a abordagem por processos na construção do modelo de simulação.
- ECSL – é uma linguagem que usa a abordagem por actividades para a construção do modelo.
- DYNAMO – desenvolvida pela *M.I.T. Computation Center* para a simulação de modelos matemáticos.
- MODSIM II – desenvolvida pela *CACI Produts Company*, é uma linguagem orientada ao objecto.
- SIMAN – introduzida em 1982, foi desenvolvida por C. Dennis Pedgen, professor na *Pennsylvania State University*.
- Simple++ – é uma linguagem de simulação orientada ao objecto; permite projectar, simular e visualizar sistemas de produção, isto é, faculta a criação de modelos e

respectiva simulação, para os diferentes níveis hierárquicos existentes num ambiente fabril. É um produto da *AESOP Corporation*.

- ARENA – é uma linguagem de simulação orientada ao objecto, desenvolvida tomando por base a linguagem SIMAN. É um produto da *Rockwell Software*.
- AnyLogic - é outro pacote de simulação com carácter genérico, com aplicações nos domínios de processo de negócios, manufactura, logística, cadeias de abastecimento, tráfego, dinâmica social, entre outros, e é baseado na linguagem Java. Este software será objecto de uma apresentação detalhada no capítulo seguinte.

Pacotes de simulação específicos

Os pacotes de software apresentados nesta secção foram projectados para aplicações específicas, no âmbito do seu domínio de aplicação, tais como: produção, saúde pública, etc.

São exemplos de sistemas deste tipo:

- SIMFACTORY – desenvolvido para a simulação de instalações industriais.
- WITNESS – é um pacote desenvolvido, em 1986, pela *ISTEL*, depois *AT&T ISTEL*, para a simulação de sistemas de produção.
- AWARD – desenvolvido por Brito em 1992, é um sistema de apoio à decisão de projecto e operação de armazéns.

Capítulo 3

Apresentação do software AnyLogic

3.1 - Introdução

Neste capítulo será apresentada a ferramenta de simulação AnyLogic, a mesma onde será desenvolvida a implementação do caso de estudo que se irá tratar no capítulo quatro.

Numa visita ao site oficial da XJ Technologies é possível conhecer o produto AnyLogic.

O AnyLogic é a primeira e única ferramenta de simulação a reunir numa só linguagem de modelação e num só ambiente de desenvolvimento, a abordagem de sistemas dinâmicos, a abordagem de eventos discretos e a baseada em agentes.

O AnyLogic é baseado em Java e na Eclipse Framework que tem sido adaptado por várias empresas. Graças ao Eclipse é possível correr o AnyLogic em vários sistemas operativos (Windows, Mac, Linux).

O Anylogic dispõe de uma basta gama de objectos de análise de dados e de gráficos, pensados para, processar eficazmente e visualizar dinamicamente a evolução dos dados durante a simulação.

A linguagem usada para a construção algoritmos, estrutura de dados, e conectividade externa é o Java. Se necessário, pode ser construídos partes de código adicionais, o que conduz a uma flexibilidade virtual ilimitada. O Java torna os modelos do Anylogic multi-plataforma, mais ainda, podem ser publicados como *applets* e ser executados remotamente através de um Web *browser*.

3.2 - Modelação sistemas a eventos discretos

O mundo que nos rodeia parece ser contínuo, pois a grande maioria dos processos que observamos encontram-se em mudança constante. Contudo, quando se estuda esse mesmos processos, na maioria dos casos faz sentido abstrairmo-nos da sua natureza contínua e considerar apenas os momentos mais importantes, os eventos que ocorrem durante o tempo execução do sistema. O paradigma de modelação que sugere a aproximação de processos reais por eventos tem o nome de modelação a eventos discretos.

Alguns exemplos de eventos: um cliente que chega a uma loja, um camião que acaba de carregar, um tapete rolante que pára, o lançamento de um novo produto, um inventário que chega a um determinado ponto, etc. Na modelação sistemas a eventos discretos o movimento, por exemplo de um comboio de um ponto A para um ponto B deve ser modelado por dois eventos: a partida e a chegada, e o movimento propriamente dito será o tempo entre estes dois eventos. O que não significa que não seja possível animar o movimento, na realidade o AnyLogic tem a capacidade de produzir animações visuais contínuas para modelos discretos.

Um evento no Anylogic tem um tempo de execução nulo, não interfere com outros eventos existentes, pode alterar o modelo quando executado e pode ele próprio agendar outros eventos. Se o motor do AnyLogic estiver a executar um modelo puramente discreto, o tempo é essencialmente uma sequência de eventos, e o motor apenas salta de um evento para o outro.

Se existirem eventos agendados para ocorrer ao mesmo tempo, serão ordenados e executados um após o outro seguindo uma sequência interna.

3.3 - Modelação de sistemas baseados em agentes

Embora seja possível encontrar várias definições para modelação baseada em agentes, no ponto de vista de aplicações práticas a modelação de sistemas baseados em agentes pode ser definida simplesmente como método descentralizado.

Na realização de um modelo baseado em agentes identifica-se as entidades activas, os agentes (que podem ser pessoas, empresas, projectos, veículos, cidades, animais, produtos, etc), define-se o seu comportamento, colocam-se num determinado ambiente, podem ser estabelecidas ligações entre eles e corre-se a simulação. O comportamento global surge como resultado das interacções entre os diferentes comportamentos individuais. O AnyLogic suporta modelação baseada em agentes assim como permite o funcionamento em conjunto com a modelação de sistemas a eventos discretos e com a modelação de dinâmica de sistemas.

3.4 - Modelação da dinâmica de sistemas

A modelação da dinâmica sistemas centra-se no estudo do *feedback* dos sistemas. AnyLogic suporta concepção e simulação de sistemas com feedback de tal forma que é possível:

- Definir variáveis stocks e de fluxos uma a uma.
- Definir auxiliares variáveis.
- Usar funções tabela com a passo, linear ou interpolação.
- Definir *arrays* de variáveis com número arbitrário de dimensões.
- Usar várias fórmulas para diferentes variáveis do *array*.
- Usar tanto funções específicas da modelação de sistemas dinâmicos como funções em Java.

Uma das características dos diagramas causais do AnyLogic é a dependência das setas são sempre sincronizados com as fórmulas: a dependência seta de A para B será exibido automaticamente quando se escreve A numa fórmula de B, e desaparecerá quando se excluir A. O mesmo acontece no sentido inverso: se for excluída a seta, serão excluídas da fórmula o A e o B.

Durante ou após o modelo correr é possível visualizar os valores correntes das variáveis directamente.

3.5 - Active object

Os *Active objects* são os principais blocos da construção de modelos do Anylogic. Os *Active Objects* podem ser usados para modelizar diversos objectos do mundo real, tais como: recursos, pessoas, hardware, objectos físicos, controladores entre outros.

Os dados dos *active objects* podem ser especificados por intermédio de parâmetros e variáveis.

Parâmetros são normalmente usados para parametrizar o objecto. São necessários quando as instâncias dos *active objects* têm o mesmo comportamento mas diferem no valor dos parâmetros. Todos os parâmetros têm a possibilidade de ser alterados durante a execução da simulação.

Os parâmetros dos *active objects* podem ser ligados a parâmetros dos *active objects* encasulados. Neste caso as alterações nos parâmetros propagam-se ao longo da árvore de dependências de parâmetros.

As variáveis podem ser de vários tipos, ser partilhadas por vários objectos, aparecer em diferentes equações algébricas e diferenciais e ainda podem ser alteradas de uma forma contínua ao longo do tempo.

Os comportamentos dos *active objects* podem ser do tipo discreto, contínuo ou então híbrido.

- Os processos contínuos podem ser descritos por equações diferenciais e algébricas.
- As actividades discretas dentro de um objecto podem ser definidas usando eventos em casos mais simples ou *statecharts* para casos mais complexos.
- Quando os comportamentos discretos e contínuos são interdependentes necessitam de uma modelação híbrida a qual pode ser conseguida através de *statecharts*.

A título de exemplo, será agora explicado a forma de implementar uma função num *active object* e uma java class no AnyLogic, os exemplos serão feitos com o recurso a implantação realizada no desenvolvimento do sistema.

Na figura 3.1 vê-se o painel geral da implementação de uma função. Neste painel são definidos o nome, o tipo de acesso, o tipo de retorno e os argumentos. O tipo de acesso pode ser, público, privado ou protegido, por defeito é considerado do tipo público.

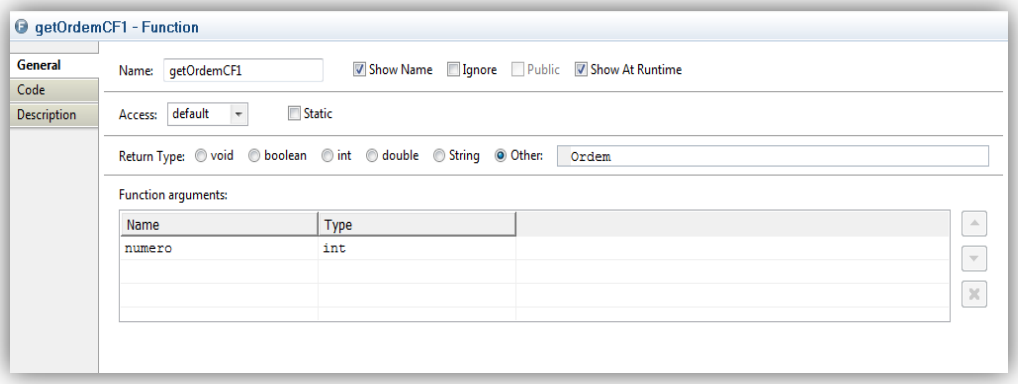

Figura 3.1 - Painel geral da edição de funções

A figura 3.2 mostra o local destinado à edição do código da função.

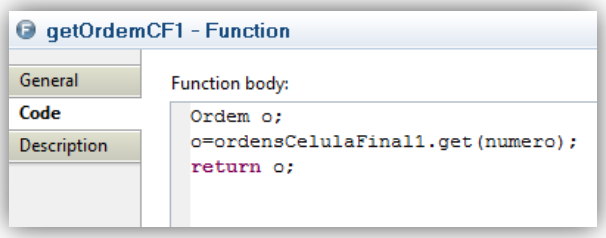

Figura 3.2 - Painel código de edição de funções

A criação de uma java class, como toda a lógica de implantação no AnyLogic é simplificada por ajuda visual.

Após a escolha de inserção de uma nova java class é mostrada a figura 3.3. Segue-se então a escolha do nome, e caso desejado, uma superclass a que pertença. O passo seguinte é a inclusão dos campos da classe. No caso da classe *Componente* os campos são a referência e a quantidade, o construtor e o método *toString* são gerados automaticamente, se assim for desejado. O método *toString* tem uma importante função na animação da simulação, pois permite que seja possível ser visualizado o conteúdo dos campos da classe.

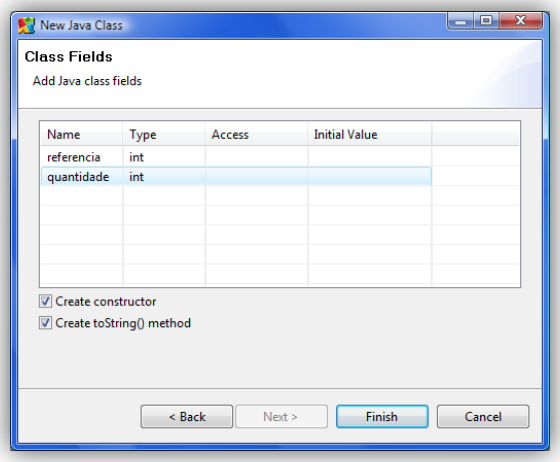

Figura 3.3 - Edição dos campos de uma java *class*

A figura 3.4 mostra o código gerado automaticamente pelos passos anteriormente descritos. Como é natural este código é passível de ser alterado directamente.

Componente.java

```
/**
 *  Componente
 */
public class Componente extends Entity {

     int referencia;

     int quantidade;


    /**
     * Default constructor
     */
    public Componente(){
    }

    /**
     * Constructor initializing the fields
     */
    public Componente(int referencia, int quantidade){
        this.referencia = referencia;
        this.quantidade = quantidade;
    }

    @Override
    public String toString() {
        return
            "referencia = " + referencia +" " +
            "quantidade = " + quantidade +" ";
    }

}
```

Figura 3.4 - Codigo da java *class* Componente

3.6 - AnyLogic Enterprise Library

A *Enterprise Library* suporta a modelação centrada no processo. Usando os objectos da *Enterprise Library* é possível modelar sistemas reais em termos de entidades (clientes, produtos, componentes, veículos, etc), processos e recursos. Os processos são especificados sob a forma de fluxogramas, os quais são extensíveis e orientados ao objecto o que permite ao utilizador modelar sistemas de grande complexidade com grande nível de detalhe. Outra capacidade importante da *Enterprise Library* é a possibilidade de criar animações sofisticadas dos processos.

Contém um conjunto de objectos específicos para processos que ocorrem num determinado espaço físico e envolvem o movimento de entidade e recursos.

A *Enterprise Library* é composta por um conjunto de objectos de alto nível que permitem a rápida construção de modelos discretos num estilo de fluxogramas. Os fluxogramas são um método gráfico de representação no qual se encaixam vários domínios importantes tais como: manufactura, logística, fluxo de trabalho, serviços, processos de negócio, rede de computadores, telecomunicações, etc.

A *Enterprise Library* encerra vários objectos para os fluxogramas: *sources*, *queues*, delays, *conveyors*, etc. Com esta biblioteca é possível a criação de modelos e animação apenas por *drag-and-drop*.

Ao mesmo tempo é possível estender as funcionalidades com a inserção de código, criação de novas classes de objectos, adicionar lógica discreta, adicionar lógica continua, etc.

Figura 3.5 - Objectos constituintes da *Enterprise Library*

Alguns objectos da *Enterprise Library*:

Source - A *source* tem como função gerar entidades, e normalmente é o ponto de partida dos fluxogramas. Aqui podem ser criadas entidades de qualquer subclasse da classe *Entity*. Os momentos da geração das entidades tanto podem ser baseados numa lei distribuições ou em tempos específicos. Também pode ser definido o número entidades geradas de cada vez, assim como o número total de entidades a ser criadas.

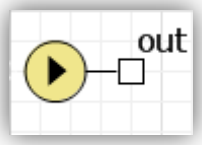

Figura 3.6 - Ícone do objecto Source

Queue - A *queue* é um buffer de entidades, em espera para ser aceites pelos objectos seguintes no fluxo do processo, pode também ser usada apenas para o armazenamento de entidades. Uma entidade pode sair da *queue* das seguintes formas:

- Normalmente saindo pelo *out* quando o próximo objecto está pronto a receber a entidade. Segue a regra FIFO.
- Pela *outTimeout* quando o tempo definido para esperar na *queue* chegou ao fim.
- Pela *outPreempted* quando a *queue* está cheia e a chegada de uma nova entidade empurra a ultima para a *outPreempted*, ou então se assim estiver definido será a própria entidade que chega a sair.
- Manualmente pela chamada do método remove da *queue*.

Pode ainda ser definido a capacidade da *queue* assim com a sua animação.

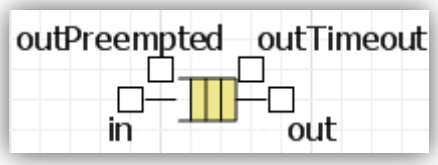

Figura 3.7 - Ícone do objecto *Queue*

Enter- Este objecto é usado, tipicamente, para transferir entidades já criadas para outro local fora do fluxo do processo. Pode também ser usada em conjunto com o objecto ***Exit*** para estabelecer rotas no modelo do processo. Para proceder à inserção de uma entidade usando este objecto utiliza-se o método interno *take*, ou então, a entidade é deixada na porta *inExternal*.

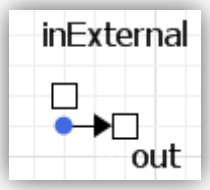

Figura 3.8 -Ícone do objecto *Enter*

Delay – O objecto *delay* pára a entidade durante um determinado tempo. Este tempo pode ser especificado dinamicamente e pode ser de natureza estocástica. Também pode ser definido com recurso a propriedades da entidade. Por exemplo o tempo de paragem pode ser proporcional ao tamanho da entidade. Existe ainda a possibilidade de definir o tempo através da animação, sendo que o tempo será determinado consoante o caminho representado pela animação e pela velocidade, definida no objecto *delay*, para percorrer esse caminho.

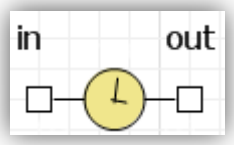

Figura 3.9 - Ícone do objecto *Delay*

SelectOutput – O *SelectOutput* aceita uma entidade, e depois encaminha-o para uma das suas portas de saída dependendo das condições especificadas. As condições podem depender da própria entidade ou então de outros critérios como por exemplo condições de probabilidade.

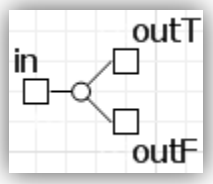

Figura 3.10 - Ícone do objecto *SelectOutput*

Sink- o objecto *sink* é o ponto final do processo. Quando chegadas a este ponto as entidades são destruídas.

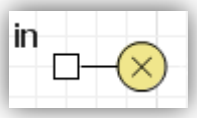

Figura 3.11 - Ícone do objecto Sink

Na figura 3.12 pode ser visto um modelo simples de fluxo. Este do modelo recria uma situação onde existem dois tipos de peças, as quais são processadas numa mesma máquina numa primeira fase e numa segunda fase cada tipo de peça é processada por máquinas diferentes.

Os dois tipos de peças são gerados nas *sourceA* e *sourceB*, ambas sofrem o seu primeiro processamento na maquina1 sendo que depois cada tipo segue um caminho diferente, uma vai para a maquina2 enquanto o outro tipo vai para a maquina1.

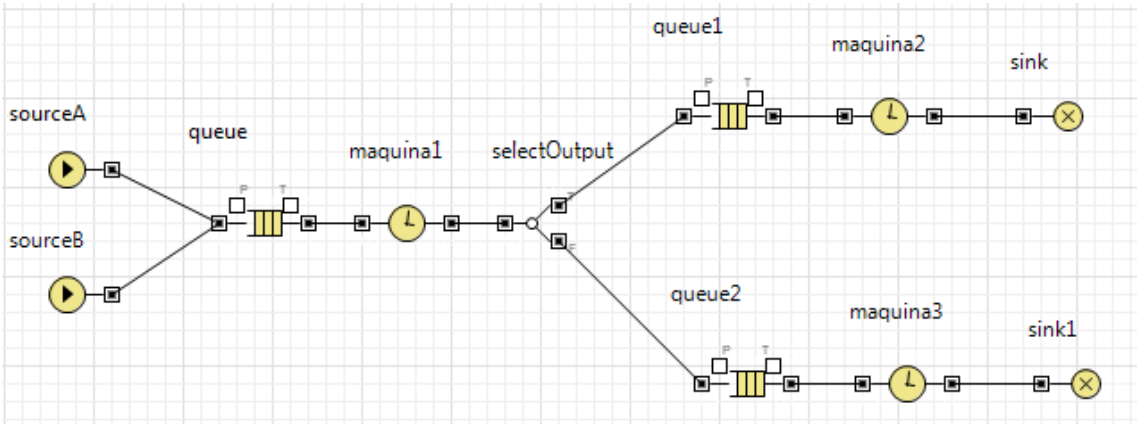

Figura 3.12 - Exemplo de modelação de processo no AnyLogic

Regras de fluxo de entidades

Todos os objectos da *Enterprise Library* têm *ports* de entrada e *ports* de saída, apenas é possível conectar saídas a entradas e vice-versa, caso contrário irá ocorrer um erro durante a execução da simulação.

O protocolo de passagem de entidades é o seguinte: o objecto que envia a entidade avisa o objecto que a recebe, ou seja o que está ligado na sua saída, que tem uma entidade pronta para sair. Se o objecto que vai receber a entidade estiver em condições de a poder receber, pede ao objecto de envio para enviar a entidade, isto se a entidade ainda lá se encontrar.

Existem dois tipos de portas de saída a *OutPort* e a *OutPortPush*. A primeira é usada em objectos que conseguem esperar que a entidade possa ser aceite pelo objecto de destino, é o caso do objecto *queue*. A maioria dos objectos apenas "empurra" a entidade para o objecto de saída. Caso a entidade não seja capaz de sair imediatamente de uma porta deste tipo irá ser gerado um erro.

Como pode ser observado no exemplo da figura 3.12 é possível estabelecer a conexão de múltiplas saídas a uma entrada, assim como o contrario também é possível.

Modelação em rede

Existem também objectos da *Enterprise Library* com o prefixo *Network* que servem de base à modelação de *layouts*. Normalmente, são usados quando os processos a ser modelados ocorrem num determinado espaço físico e incluem o movimento de entidades e de recursos. Para utilizar este conjunto de objectos é necessário definir a topologia de rede, os recursos, e o próprio processo.

Na definição do processo, pode ser feita uma mistura de objectos específicos de rede, tais como NetworkMoveTo ou *NetworkExit*, e os restantes objectos da *Enterprise Library*. A animação das entidades e dos recursos é feita automaticamente, estes deslocar-se-ão ao longo dos segmentos de rede ou permanecerão nos nós. Uma rede é um conjunto de nós interligados por segmentos.

O movimento na rede é sempre feito ao longo do caminho mais curto entre o nó de origem e o nó de destino. As entidades e os recursos podem ter velocidades individuais diferentes, e essas velocidades podem alterar dinamicamente. Por exemplo, podem ser definidas diferentes velocidades para um camião carregado e descarregado. Os segmentos têm capacidade infinita, de modo que entidades que se deslocam ao longo de um segmento não interferem umas com as outras. Podem existir múltiplas redes num modelo, e as

entidades podem passar de uma para outra, porém os recursos associados a uma rede não podem sair dessa mesma rede.

Para entrar numa rede a entidade deve passar por um objecto *NetworkEnter*, e para sair passar pelo *NetworkExit*. Para mover uma entidade na rede é utilizado o *NetworkMoveTo*.

Os recursos associados a uma rede podem ser de três tipos: estáticos, móveis ou portáteis. Os recursos estáticos estão ligados a um determinado nó dentro da rede e não se podem mover. Os recursos móveis podem mover-se por conta própria, por exemplo pessoal, veículos. Os recursos portáteis podem ser movidos pelas entidades ou por recursos móveis. Estes recursos têm a sua origem nos recursos locais e podem opcionalmente ser devolvidos ao local de origem. Os recursos são definidos com o objecto *NetworkResourcePool*.

A gestão dos recursos é feita de forma centralizada. A rede mantém a fila de pedidos para a atribuição dos recursos os quais são atendidos da forma FIFO. Um pedido pode ser satisfeito se todos os recursos solicitados estão disponíveis em simultâneo, caso contrário, os recursos que estão disponíveis serão reservados para o pedido e este permanece na fila. Isto significa que um pedido que se encontre a meio da fila só poderá ser satisfeito se não colidir com nenhum pedido na frente dele.

3.7 - Agente

Um agente, em modelação baseada em agentes, é uma unidade do modelo que contém, comportamento, memória, contactos, etc. Os agentes podem representar pessoas, empresas, projectos, veículos, cidades, animais, produtos, etc. A classe *Active Object* do AnyLogic é uma forma básica para criar agentes com as suas propriedades: num *active object* pode-se definir variáveis, eventos, máquinas de estados, sistemas dinâmicos, pode-se ainda colocar outros *active objects* existentes. Tipicamente os agentes serão introduzidos em *active objects* de um nível superior.

A criação de um agente começa pela identificação das suas variáveis principais e interface com o espaço exterior. Podem ser consideradas duas formas de estabelecer a ligação entre agentes: usando as *ports*, no caso de serem poucos agentes, ou no caso em que o número de agentes é maior as ligações devem ser criadas dinamicamente e os agentes comunicam através da chamada de métodos de cada um.

O estado e comportamento interno de um agente podem ser implementados de duas formas. O estado do agente pode ser representado por um conjunto de variáveis ou por uma máquina de estados. O comportamento interno pode ser passivo, isto é o agente apenas reage à chegada de mensagens ou à chamada de métodos, ou por outro lado é activo, quando a dinâmica interna causa a acção do agente.

3.8 - Interacção entre objectos

O AnyLogic suporta diversos mecanismos de comunicação, tanto contínuos como discretos, tais como conexão por variáveis, troca de mensagens e propagação de parâmetros.

Conexão por variáveis

A interacção entre dois objectos por ser realizada através da ligação das suas variáveis. As variáveis que estão ligadas terão sempre o mesmo valor, isto é a alteração de uma variável

será imediatamente propagada para outra variável que esteja declarada como dependente. Esta capacidade pode ser utilizada tanto em sistemas discretos como em sistemas dinâmicos.

Troca de mensagens

No AnyLogic é possível estabelecer a interacção entre objectos através da troca de mensagens. As mensagens podem modelar vários objectos do mundo real. Podem ser usadas para mecanismos de notificação ou sinalização, nestes casos podem representar comandos ou sinais passados pelo sistema de controlo. Ou então podem representar o fluxo de entidades, onde as mensagens representam as entidades. As mensagens são recebidas e enviadas através de elementos especias dos *active objects* que são as *ports*. As mensagens são passadas entre duas *ports* apenas se estas estiverem ligadas.

Propagação de parâmetros

A propagação de parâmetros consiste em associar um parâmetro de um *active object* ao parâmetro do *active object* que se pretende efectuar a ligação. Neste caso se foi efectuada alguma alteração durante a execução do modelo o parâmetro dependente também se altera. Este comportamento mantém-se, desde o ponto de modificação, para todos os parâmetros dependentes ao longo da árvore de objectos.

3.9 - Máquina de estados

Durante o seu tempo de vida, um *active object* efectua operações de resposta a eventos e condições, internos ou externos. A existência de um *statechart* num *active object* significa que a ordem pela qual as operações são chamadas é importante. Um *statechart* é usado para mapear os estados de um determinado algoritmo, os eventos que provocam a transição de um estado para outro, e as acções resultantes da mudança de estado.

O Anylogic permite a construção de *statecharts* híbridos, onde é possível associar um conjunto de equações a um estado do *statechart*. Quando uma transição é disparada como resultado de, por exemplo um evento discreto, ocorre uma alteração de um sistema de equações. Desta forma a lógica discreta afecta o comportamento contínuo. Se o disparo de uma transição tiver como condição a variação de uma determinada variável contínua, ter-se-á o comportamento contrário, ou seja o comportamento contínuo tem impacto na parte discreta do sistema.

Os *statechats* do Anylogic preservam o aspecto gráfico, os atributos e a execução semântica definida pelo UML.

Máquinas de estado são compostas por estados e transições, como pode ser visto na figura 3.13. As transições são disparadas segundo as condições definidas, as quais podem ser do tipo, mensagens recebidas pela máquina de estados, tempo, taxas, condições booleanas. A execução de uma transição pode levar a uma alteração de estado onde novas transições ficarão activas. Os estados podem ser hierárquicos, ou seja um estado pode conter vários estados e transições no seu interior.

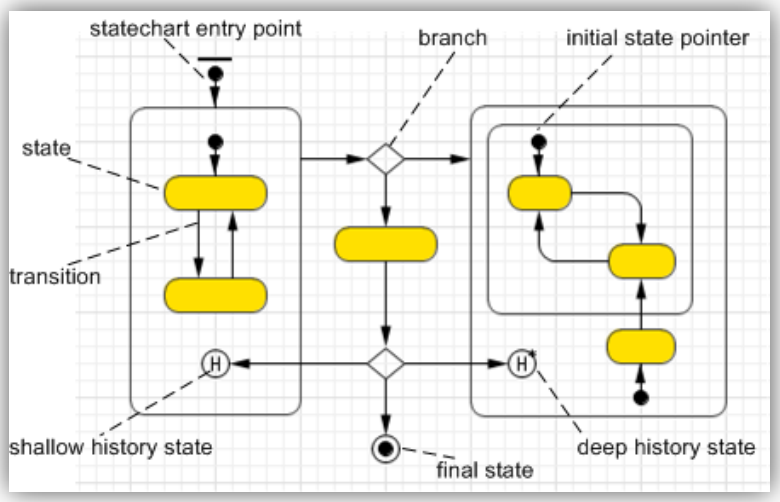

Figura 3.13 – Constituição do *Statechart*

Capítulo 4

Apresentação e modelação do sistema em estudo

4.1 - Descrição global do sistema de produção

Com o recurso à ferramenta de simulação AnyLogic, pretendeu-se desenvolver um modelo que simulasse a produção em uma célula de preparações, o fluxo de materiais produzidos nessa célula até ao local onde são consumidos, o consumo dos mesmos assim como avaliar vários factores performance, tais como o número de vezes que ocorrem roturas de abastecimentos e por consequência paragem da célula final, a taxa de ocupação dos operários, a taxa de ocupação dos postos da célula de preparações, o número de caixas produzidas nas células e o número de caixas de cada referência produzidas nas células.

Numa descrição geral do sistema de produção identifica-se os seguintes módulos:

- Supermercado
- Células de preparações
- Supermercados das células de preparações
- Células finais
- *Milk runs*

Supermercado – local onde se encontram os componentes para abastecimento das células de preparações.

Célula de preparações – local onde se produzem os componentes a serem utilizados nas células finais.

Supermercado da célula de preparações – local onde se armazenam os componentes produzidos na célula de preparações.

Célula final – local onde se produzem os produtos para entrega ao cliente.

Milk run – elemento com a responsabilidade de recolher os componentes nos supermercados das células de preparações e os entregar na célula final.

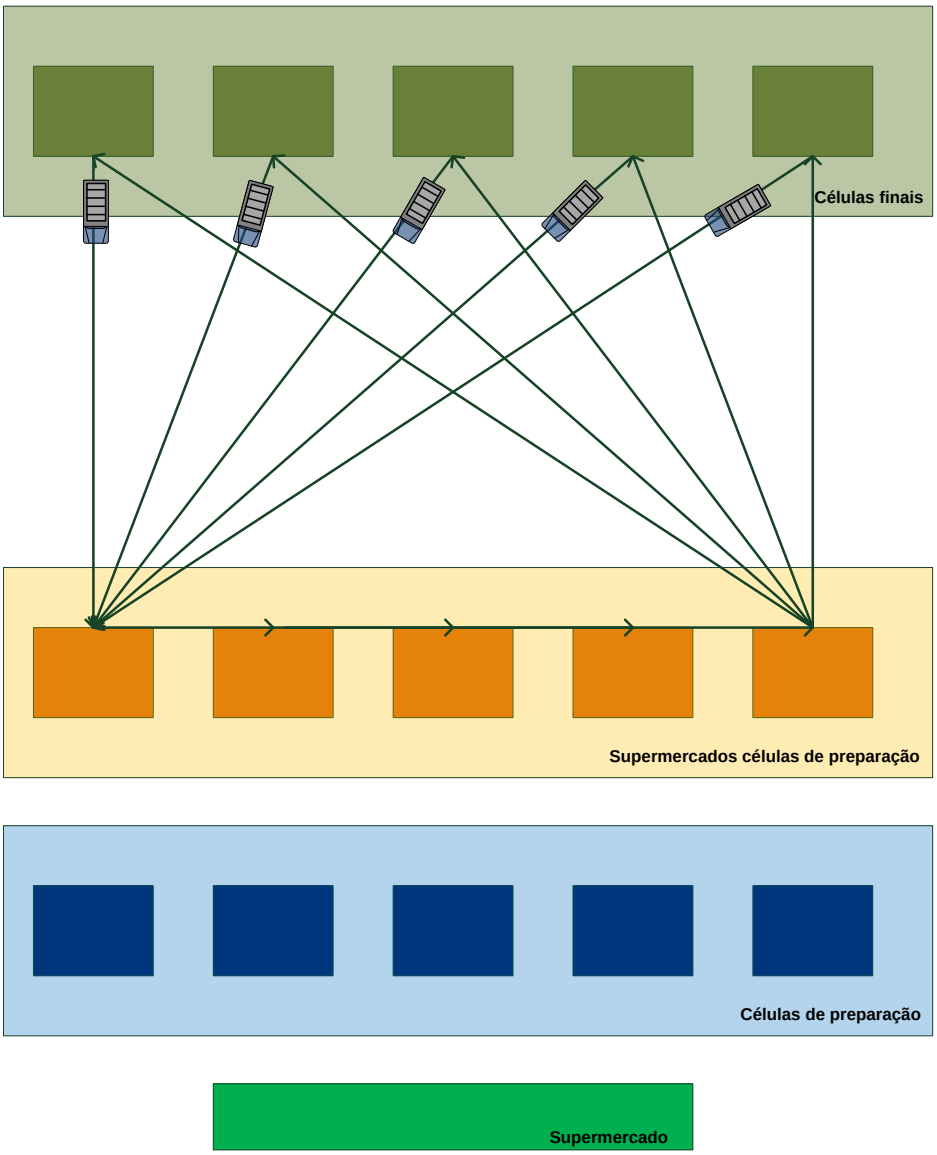

Figura 4.1 - Ilustração do sistema geral

Visto de uma forma geral o desenrolar da produção desenvolve-se da seguinte forma:

- As células finais têm um plano de produção a cumprir, sendo assim necessitam de componentes específicos para cada ordem de produção.
- Todas as ordens de produção têm uma quantidade normalizada e de valor igual dezasseis.
- Cada célula final tem um *milk run* responsável pelo seu abastecimento.
- O *milk run* percorre os supermercados das células de preparações com o objectivo de recolher os componentes necessários à célula final que abastece. Depois de recolhidos os componentes são entregues na célula final.
- A célula de preparações tem como função repor o stock de componentes no seu supermercado.
- A célula de preparações retira do supermercado os componentes necessários para produção necessária à reposição do stock no seu supermercado.

Nesta dissertação o objectivo é estudar o funcionamento de uma célula de preparações.

4.2 - Apresentação do caso de estudo

O sistema de estudo encontra-se centrado em uma célula de preparações, pois na realidade e salvo pequenos detalhes, no ponto de vista de simulação não existe grande diferença entre as células de preparações. Isto porque diferenças do género de tipo de recursos, afectação de recursos, tempos de operação são alteradas com facilidade na simulação, o que no sistema real já não funciona da mesma forma.

Visto a orientação do caso de estudo, a representação do supermercado, assim como o abastecimento da célula de preparações foi suprimido, pelo que a partir deste ponto quando se referir supermercado, está-se na realidade a referir o supermercado da célula de preparações.

Na figura 4.2 pode ser observado os elementos físicos existentes no sistema de estudo. O qual é constituído por quatro células finais, uma supermercado e quatro *milk runs*, um para cada célula final. Existe também uma célula de preparações que contém três operários e seis postos.

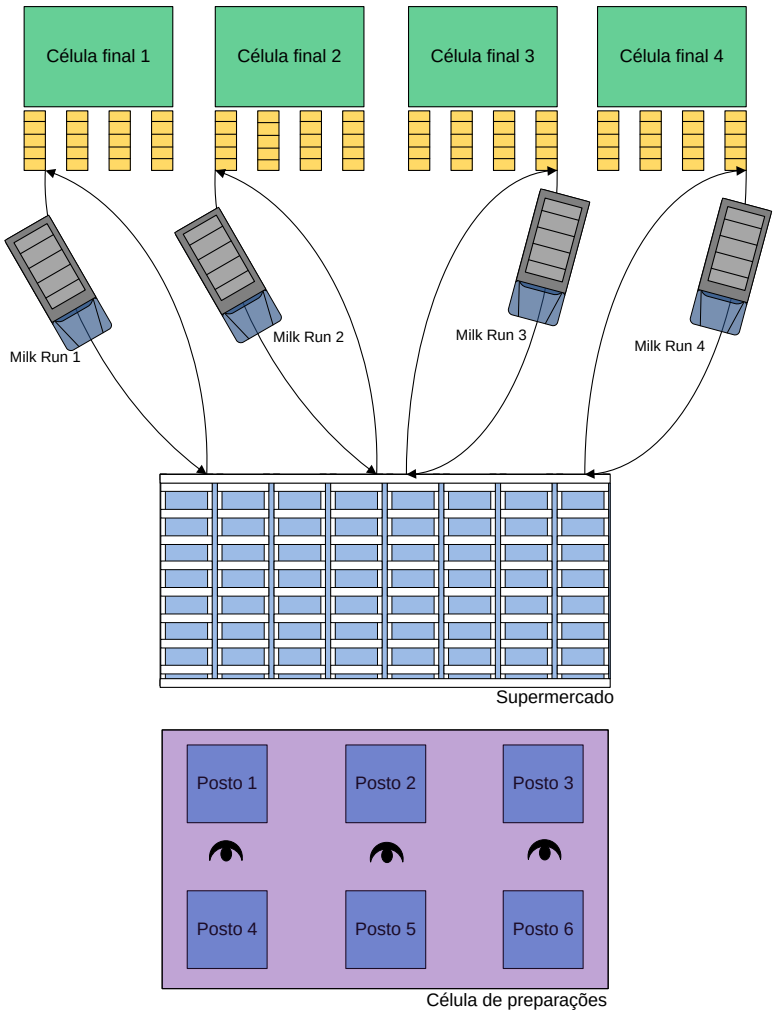

Figura 4.2 - Ilustração do sistema de estudo

4.3 - Modelação do sistema de produção

Conforme representado na figura 4.3, na modelação do sistema de produção são consideradas duas perspectivas principais:

- A modelação da estrutura.
- A modelação da dinâmica.

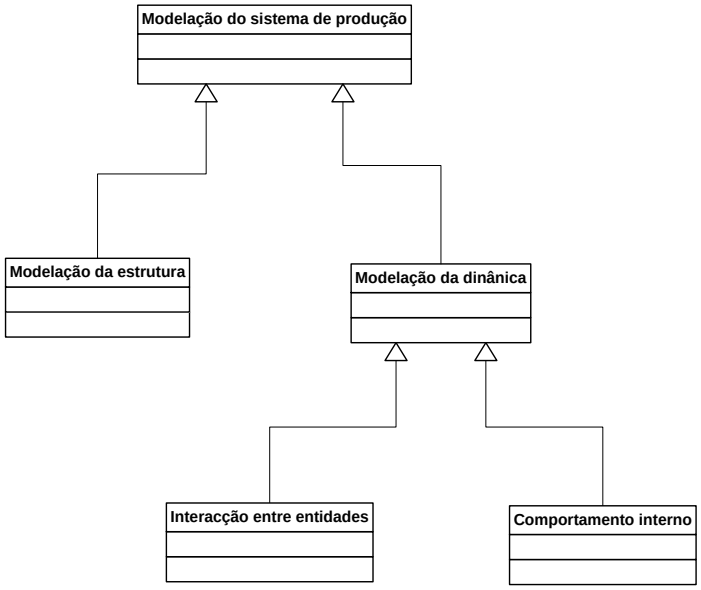

Figura 4.3 - Perspectivas da modelação do sistema

O modelo da estrutura representa os elementos constituintes do sistema e a relação entre eles, e será representado sobre a forma de um diagrama de entidades.

Na análise e modelação da dinâmica do sistema serão considerados dois níveis:

- Um nível global.
- Um nível local.

Ao nível global são onde são representadas as interacções entre entidades que constituem o sistema, através de diagramas de sequência.

Um nível entidade onde é representado o comportamento interno de cada entidade através de modelos do tipo diagramas de estado.

Modelo da estrutura do sistema de produção

O diagrama de entidades que constituem a estrutura do sistema está representado na figura 4.4 e tem como objectivo descrever as várias entidades do sistema e o seu comportamento para assim melhor compreender o funcionamento do sistema de produção e servir de base à elaboração do modelo de simulação. De facto, como se verá no capítulo 5, há uma correspondência directa entre os elementos do modelo UML e os elementos do modelo de simulação.

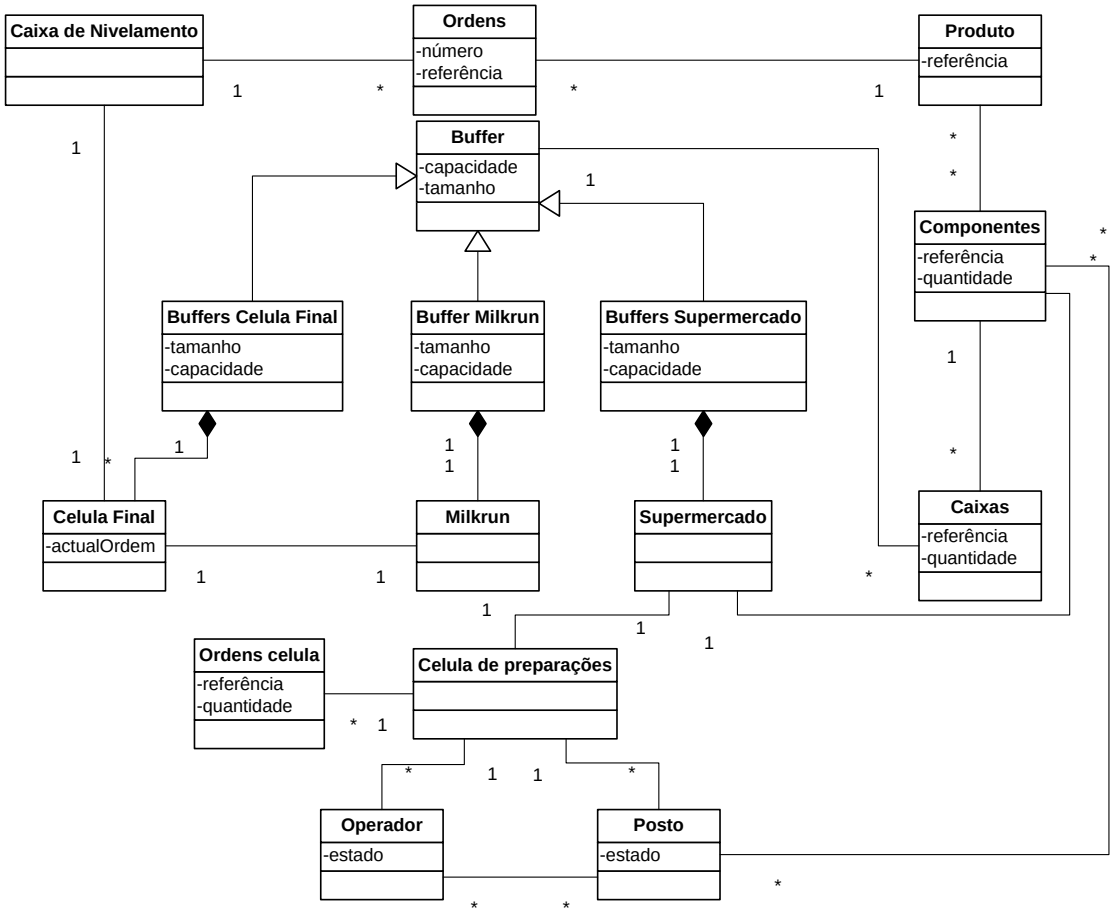

Figura 4.4 - Diagrama de entidades

A entidade buffer é a responsável pelo armazenamento de todas a caixas do sistema de produção, pelo que a mesma terá obrigatoriamente que estar ligada a todas as entidades onde existem o armazenamento de caixas. Será, portanto, necessário saber a sua capacidade e quantidade de caixas que contém. Caixas representam as caixas físicas que contêm os produtos e componentes.

Sendo que existe armazenamento tanto na célula final, nos *buffers* de bordo de linha, como no supermercado, é natural a ligação entre estes e o *buffers*, já no diz respeito à ligação entre os *buffers* e o *milk run* é explicada pelo facto do armazenamento durante o tempo de transporte das caixas do supermercado para a célula final e vice-versa. Os buffers de bordo de linha representam as posições situadas em torno das células finais as quais contêm as caixas com os componentes utilizados na montagem do produto final.

A célula final é responsável pela produção dos produtos do sistema de produção, para que haja o conhecimento da parte da célula final de quais as ordens que deve produzir é estabelecida uma ligação desta à caixa de nivelamento. A qual, por sua vez, se liga às ordens propriamente ditas. As ordens contêm a referência e o número de sequência pela qual devem ser produzidas.

Os produtos produzidos no sistema de produção são referidos nas ordens, já a sua constituição encontra-se descrita através da ligação deste aos respectivos componentes constituintes. Para além de se ficar a saber qual a referência dos componentes constituintes saber-se-á também em que quantidade eles são necessários.

As caixas que circulam nos *buffers* do sistema têm no seu interior uma determinada quantidade de componentes duma certa referência.

A célula de preparações tem a função de produzir as caixas a ser armazenadas no supermercado. Daí a ligação às ordens da célula, para saber qual a referência e quantidade a produzir. O acto da produção está a cargo dos posto e dos operários dos quais se sabe qual o estado presente.

Modelação da dinâmica

Interacção entre entidades

O funcionamento do sistema de produção envolve três ciclos de operação principais.

- Ciclo de execução de ordem de produção na célula final.
- Ciclo de abastecimento da célula final a partir dos supermercados.
- Ciclo de produção na célula de preparação.

Os quais são descritos de seguida através do respectivos diagramas de sequência.

Ciclo de execução de ordem de produção na célula final

As entidades envolvidas no ciclo estão representadas na figura 4.5:

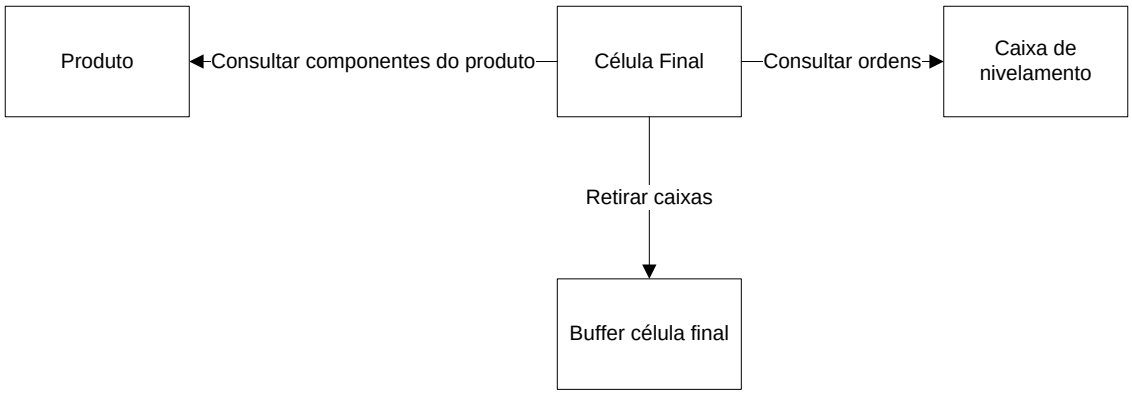

Figura 4.5 - Entidades envolvidas no ciclo de produção na célula final

O diagrama de sequência que representa as interacções entre as entidades está representado na figura 4.6.

O ciclo envolve as interacções seguintes:

- A célula final começa por efectuar uma consulta à caixa de nivelamento para obter a próxima ordem a produzir.
- Uma vez obtida a referência do próximo produto a produzir, a célula final consulta a entidade produto para obter os componentes necessários ao seu fabrico.
- Com base nesta informação, a célula final verifica se existem nos buffers de bordo de linha e as quantidades necessárias.
- Se as quantidades necessárias existirem retira as caixas do bordo de linha e inicia a produção da ordem.

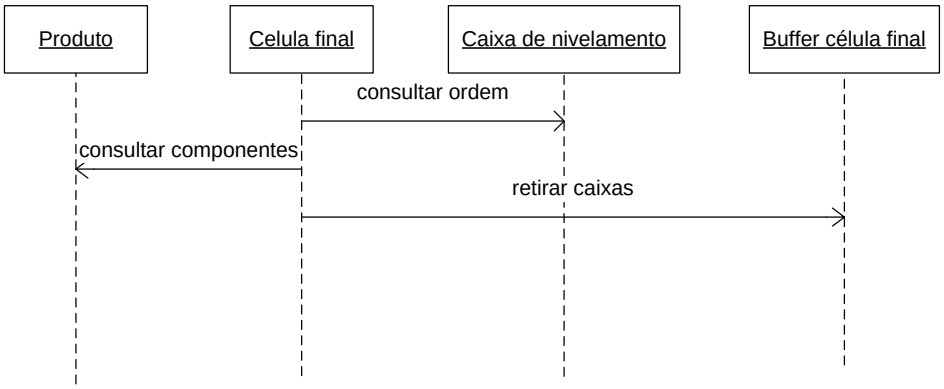

Figura 4.6 - Diagrama de interacções do ciclo de produção na célula final

Ciclo de abastecimento da célula final

Entidades envolvidas no ciclo de abastecimento da célula final a partir dos supermercados estão representadas na figura 4.7:

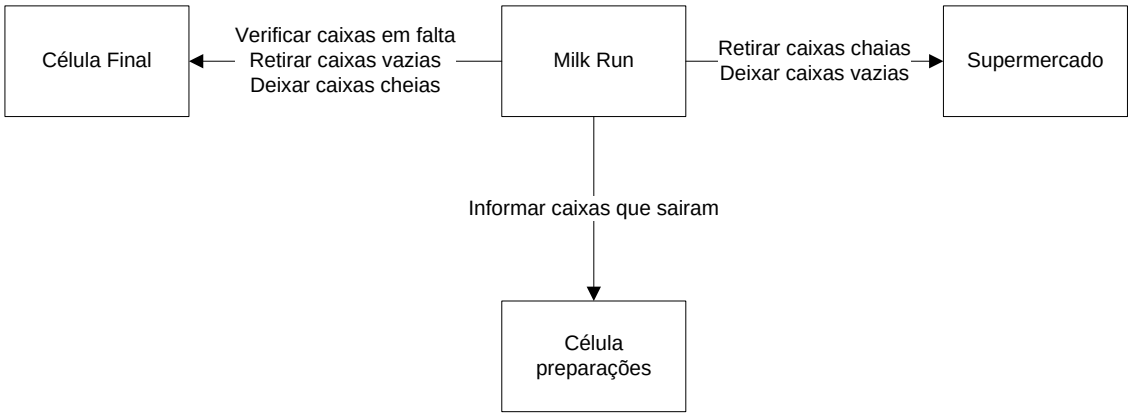

Figura 4.7 - Entidades envolvidas no ciclo de abastecimento

Diagrama de interacções entre as entidades envolvidas no ciclo de abastecimento da célula final encontra-se representado da figura 4.8

O ciclo envolve as interacções seguintes:

- Após a chegada do *milk run* à célula final, retira as caixas vazias presentes na célula.
- Ainda na célula final, o *milk run* deixa as caixas cheias que transporta para a célula final.
- A última tarefa a ser executada pelo *milk run* na célula final é a verificação das caixas que faltam, para produzir as próximas ordens.
- Quando o *milk run* se encontra no supermercado deixa as caixas vazias que recolheu na célula final.
- Também quando está no supermercado, o *milk run* retira a caixas que estão em falta na célula final.
- Por fim o *milk run* informa a célula de preparações quais as caixas que retirou do supermercado.

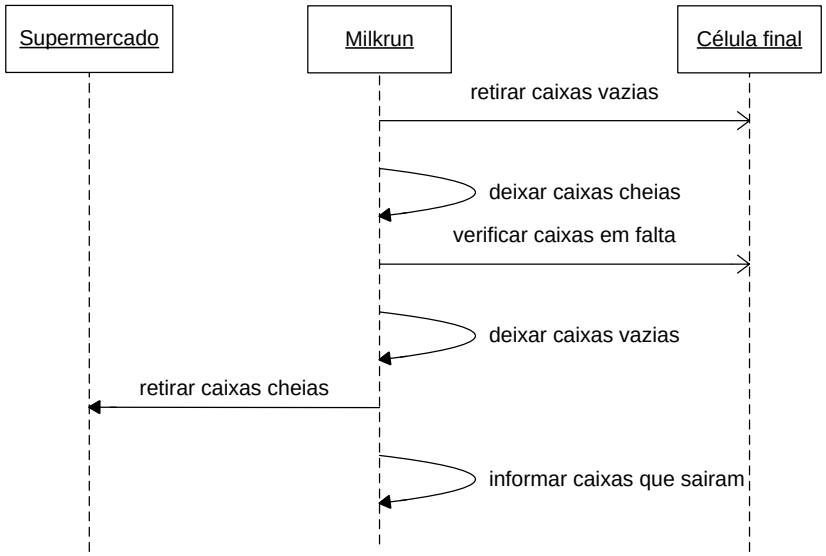

Figura 4.8 - Diagrama de interacções do ciclo de abastecimento

Ciclo de produção na célula de preparação

Entidades envolvidas no ciclo de produção na célula de preparação estão representadas na figura 4.9:

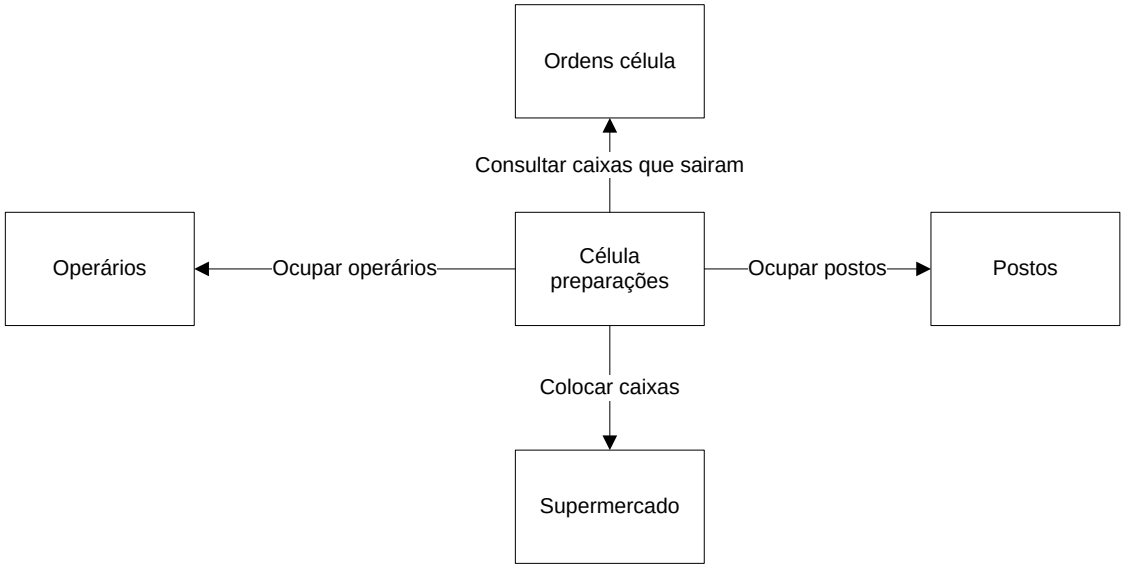

Figura 4.9 - Entidades envolvidas no ciclo de produção na célula de preparações

Diagrama de interacções entre as entidades envolvidas no ciclo de produção na célula de preparação está representado na figura 4.10.

O ciclo envolve as interacções seguintes:

A primeira interacção a ser efectuada é uma consulta por parte da célula de preparações à entidade ordens célula com a finalidade de saber qual a próxima ordem a produzir.

- Com a informação da ordem a produzir, surge então a necessidade de se obter um operário livre para executar a ordem.
- Tendo agora uma ordem a executar e quem a execute falta apenas o local onde a executar, segue-se a interacção com posto para verificar se está livre.
- Quando todas as condições anteriores forem reunidas e se der a produção, é chegada a ultima interacção, que consiste em colocar as caixas produzidas no supermercado.

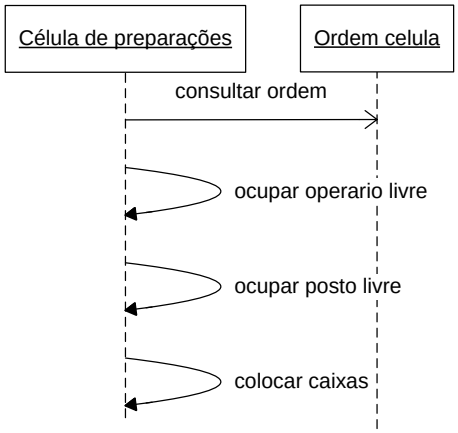

Figura 4.10 - Diagrama de interacções no ciclo de produção na célula de preparações

Modelação do comportamento interno das entidades

A dinâmica interna é descrita através de diagrama de estados de cada entidade.

Diagrama de estados da célula final

A célula final no decorrer do seu funcionamento pode estar em três estados distintos, *livre*, *falta caixa* ou *operação*.

Se a célula está no estado *livre* e surgir uma nova ordem, podem se tornar activos:

- O estado *operação*, na condição de existir nos *buffers* caixas com os componentes e quantidade necessária para executar essa ordem.
- O estado *falta caixa* se, pelo contrário, não existirem essas caixas.
- Estado *falta caixa*, permanecerá activo até que existam as caixas de componentes necessárias, e só depois de tal condição se verificar passará para o estado *operação*.

No estado *operação* são retiradas dos *buffers* as caixas com os componentes. A saída deste estado para o estado livre dar-se-á no fim da operação.

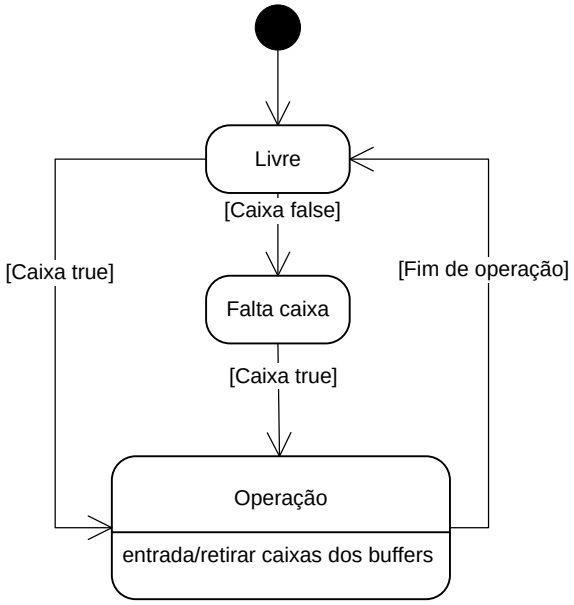

Figura 4.11 - Diagrama de estados da célula final

Diagrama de estados do posto.

Cada posto da célula de preparações tem dois estados possíveis, o estado *livre* e o estado *ocupado*.

A transição do estado *livre* para o estado *ocupado* é disparado por uma mensagem *ocupar*, enviada pela célula de preparações, enquanto a transição do estado *ocupado* para o estado *livre* é disparada no fim da operação.

Na saída do estado *ocupado* para o estado *livre*, são colocadas no supermercado as caixas produzidas.

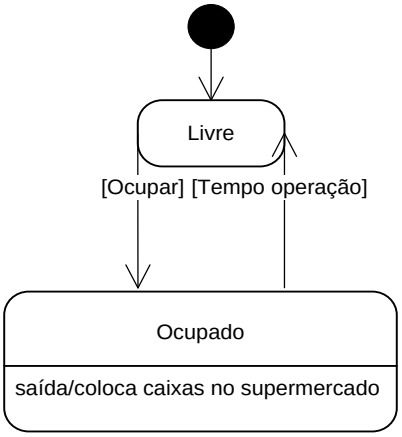

Figura 4.12 - Diagrama de estados do posto

Diagrama de estados do operário.

Os diagramas de estados dos operários são constituídos pelo estado *livre* e pelo estado *ocupado*. A passagem entre estes dois estados é controlada por mensagens da célula de preparações, e mensagens dos postos. Sendo que a célula de preparações envia a mensagem *ocupar*, enquanto a transição do estado livre para o estado *ocupado* é disparada pela mensagem *fim operação*, enviada pelo posto.

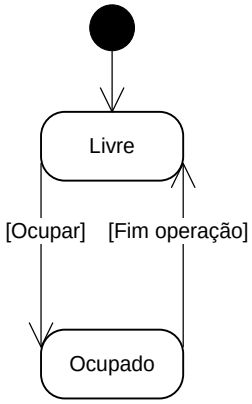

Figura 4.13 - Diagrama de estados do operário

Diagrama de estados do *milk run*.

O *milk run* irá efectuar o trajecto entre a célula final e o supermercado, logo terá os estados *mover para célula final* e *mover para supermercado*.

Estando o *milk run* na célula final, ou seja estado *na célula final*, executa tarefas como, retirar caixas vazias, deixar caixas cheias e verificar quais as caixas em falta.

Quando no estado *no supermercado* as tarefas a cumprir são, deixar caixas vazias, retirar caixas cheias e informar a célula de preparações quais as caixas que foram retiradas.

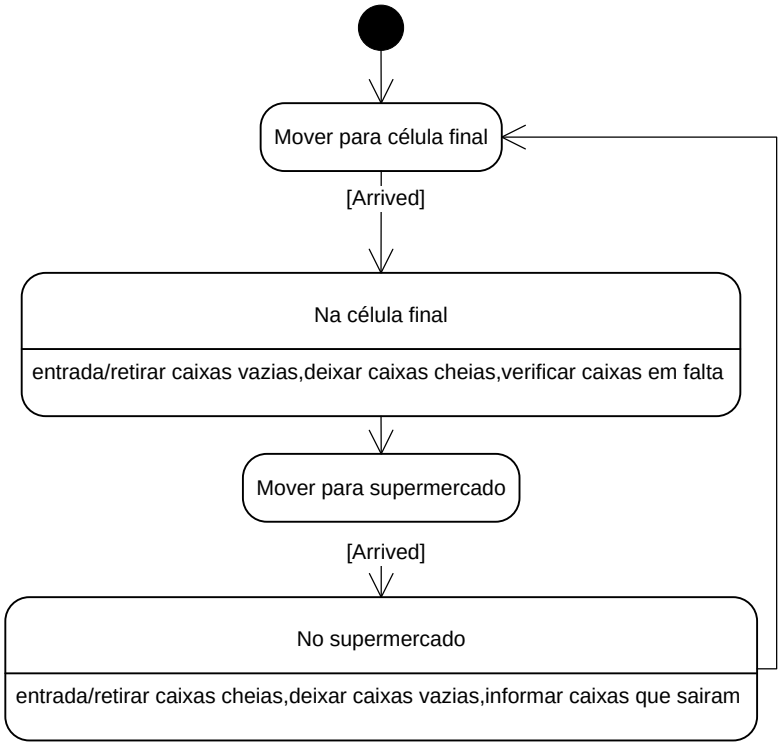

Figura 4.14 - Diagrama de estados do *milk run*

4.4 - Entidades

Buffer

A função do *buffer* é armazenar, o que leva à necessidade de métodos como *enter* e *remove* para introduzir e retirar as caixas.

No que diz respeito aos atributos, tem-se a *capacidade*, que indica qual o número máximo de caixas que podem estar presentes no buffer e ainda o *tamanho*, informa qual o número de caixas que estão no *buffer*.

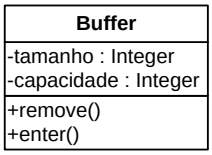

Figura 4.15 - Entidade *buffer*

Célula final

De acordo com o modelo da estrutura cada célula final estão associada a um determinado número de *buffers*, assim como ao *produto*, *caixa de nivelamento* e ao *milk run*.

De acordo com o diagrama de interacção e diagrama estados, a célula final tem na sua constituição os seguintes métodos, *verificar caixas em falta*, *retirar caixas vazias* e ainda *retirar caixas cheias dos buffers*

Os métodos *verificar caixas em falta*, e *retirar caixas cheias* são invocados pelo *milk run*, quando este se desloca até célula final.

A função *verificar caixas em falta* tem o seguinte algoritmo:

- Consultar as três próximas ordens a fabricar.
- Para cada ordem, consultar os componentes necessários para a produção.
- Verificar nos buffers se estão presentes as caixas com os componentes necessários assim como as quantidades indicadas.
- Quando, para uma determinada caixa, as condições anteriores não são conseguidas, essa caixa é acrescentada à *listaEmFalta*.

Algoritmo da função *retirar caixas cheias dos buffers*:

- Obter a ordem pretendida.
- Identificar os componentes do produto referido na ordem.
- Reconhecer qual o *buffer* associado a cada referência dos componentes.
- Percorrer cada *buffer* com a finalidade de identificar e remover as caixas com os componentes necessários.

A função *retirar caixas vazias* terá um algoritmo simples, pois apenas tem que retirar todas as caixas vazias para o *buffer* do *milk run*.

A célula final dispõe ainda das seguintes propriedades internas:

- *listaEmFalta* – lista de caixas que se encontram em falta para a produção.
- *actual* – número de ordem que a célula está a produzir.
- *taxaDeRotura* – percentagem de tempo em que a célula esteve com falta de caixas.
- *numeroRoturas* – número de vezes que ocorreu a falta de caixas.

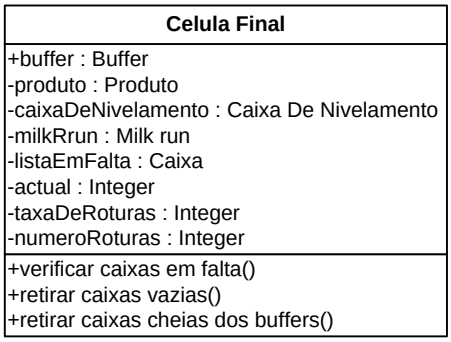

Figura 4.16 - Entidade célula final

Milk run

O *milk run* encontra-se ligado a um *buffer* no qual transporta as caixas da célula de preparações para a *célula final*, ao qual está associada.

Visto que o *milk run* tem que se abastecer está também associado ao *supermercado,* assim como à *célula de preparações* para a informar quais as caixas que saíram.

Em conformidade com o apresentado no diagrama de interacção e no diagrama de estados tem-se como métodos do *milk run*, *deixar caixas vazias*, *deixar caixas cheias*, *informar caixas que saíram*.

Os algoritmos das funções *deixar caixas vazias* e *deixar caixas cheias* são muito parecidos, ambos terão de retirar todas as caixas do *buffer* do *milk run*, as diferenças estão, naturalmente no conteúdo das caixas e no local para onde serão encaminhadas as caixas, sendo que as cheias são colocadas na célula final, enquanto as vazias irão para o supermercado.

O comportamento da função *informar caixas que saíram* consiste em, inserir, para cada caixa retirada do supermercado, uma ordem igual nas ordens da célula de preparação.

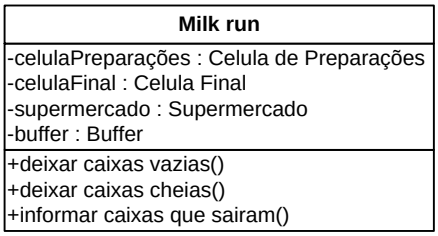

Figura 4.17 - Entidade *Milk run*

Supermercado

Sendo o supermercado apenas um local de armazenamento está a associado a *buffers*, os quais terão no seu interior caixas apenas de uma referência.

O supermercado dispõe apenas de um método, *o retirar caixas cheias*, e é invocado pelo *milk run* quando este se desloca até ao supermercado.

Algoritmo da função *retirar caixas cheias*:

- Identificar o buffer responsável por armazenar a referência pretendida.
- Percorrer o buffer e remover as caixas necessárias até perfazer a quantidade pretendida.

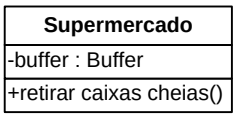

Figura 4.18 - Entidade supermercado

Caixa de nivelamento

A caixa de nivelamento é a entidade onde ficam guardadas as ordens de produção, e para que seja possível aceder a essas ordens dispõe do método *consultar ordem*.

As ordens serão guardadas num atributo interno da caixa de nivelamento, que será o *listaOrdens*.

O algoritmo do método consultar ordem, consiste em retornar a ordem, que está na *listaOrdens* com o número pedido pelo método.

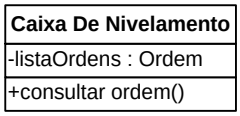

Figura 4.19 - Entidade caixa de nivelamento

Ordem

A ordem tem apenas dois atributos, o atributo número, que indica qual a posição em que a ordem se encontra na sequência de produção, e o atributo referência correspondente a referência do produto a fabricar.

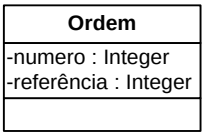

Figura 4.20 - Entidade Ordem

Produto

A entidade produto e constituída pelo atributo *listaProduto*, a qual tem no seu interior os componentes que fazem parte do produto, e também, o método *consulta componentes,* que quando invocado retorna a lista de componentes do produto pedido.

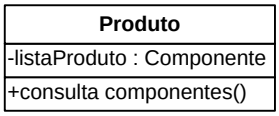

Figura 4.21 - Entidade produto

Componente

O componente tem apenas atributos, sendo estes, a *referência* e a *quantidade* necessária para a produção de uma ordem de um determinado produto que contenha este componente.

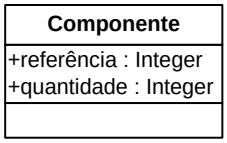

Figura 4.22 - Entidade Componente

Caixa

Cada caixa transporta uma única *referência* e uma *quantidade* que será sempre um múltiplo ou submúltiplo de 16, visto a quantidade por ordem ser normalizado e sempre igual a 16.

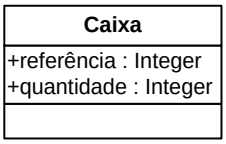

Figura 4.23 - Entidade Caixa

Célula de preparações

A célula de preparações está associada a *postos, operários* assim como a *ordem*.

Tal como foi descrito no diagrama de interacções da célula de preparações, os métodos disponíveis são o *colocar caixa*, *ocupar posto livre*, *ocupar operário livre*.

A célula de preparações dispõe ainda do atributo interno *quantidadesProduzidas*, número de caixas feitas de cada referência.

A função ocupar operário livre tem o seguinte algoritmo:

- Consultar a ordem a produzir.
- Verificar qual o posto capaz de produzir a referência do componente pedido.
- Verificar o estado em que se encontra o posto.
- Quando as condições anteriores forem validadas, colocar o posto no estado ocupado.

O algoritmo da função ocupar posto livre difere do algoritmo anterior apenas no segundo ponto, visto que os operários podem produzir qualquer tipo de componentes.

O método colocar caixa apenas tem de verificar qual o buffer do supermercado associado à referência do componentes contidos na caixa e introduzir a caixa nesse *buffer*.

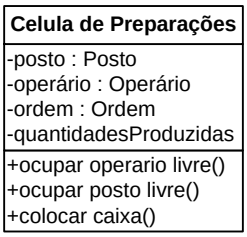

Figura 4.24 - Entidade célula de preparações

Posto

O posto é o local de produção da célula de preparações e tem como atributos:

- *tempoOperação* – tempo necessário para a produção de determinado componente.
- *caixa* – informa a quantidade e referência do componente em produção.
- *operário* – indica qual dos operários está no posto a trabalhar.

- *livre* – estado em que se encontra o posto.
- *numeroCaixasfeitas* – quantidade de caixas produzidas no posto.
- *taxaUtilização* – percentagem de tempo que o posto se encontra em produção.
- *referênciaProdução* - indica quais as referências dos componentes capaz de fabricar.

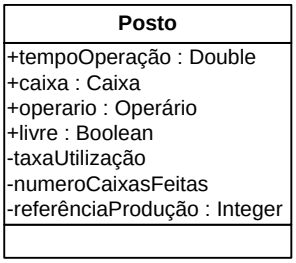

Figura 4.25 - Entidade posto

Operário

O operário executa o seu trabalho num posto da célula de preparações, logo tem um atributo que o associa ao *posto*. Tem também o atributo *livre,* que indica o estado em que o operário se encontra e a *taxaUtlização*, percentagem do tempo que o operário se encontra a trabalhar.

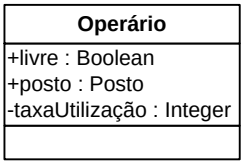

Figura 4.26 - Entidade operário

Ordem célula

A ordem célula tem apenas os atributos, *quantidade* e *referência* do componente a fabricar.

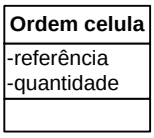

Figura 4.27 - Entidade Ordem célula

4.5 - Indicadores de desempenho

O conhecimento profundo de todas actividades de um sistema é fundamental para que seja possível atingir um rendimento máximo. De tal modo é necessário recolher informações que permitam a avaliação do rendimento.

Os principais indicadores a ter em conta devem ser o número de vezes que ocorrem roturas de abastecimentos e por consequência paragem da célula final, taxa de ocupação dos operários, taxa de ocupação dos postos da célula de preparações, número de caixas produzidas nas células e número de caixas de cada referência produzidas nas células.

Para a avaliação destes indicadores de desempenho devem ser considerados diferentes cenários tais como:

- Dimensionamento dos buffers do supermercado, ou seja a quantidade média de caixas que cada buffer do supermercado deverá conter.
- Número de operadores da célula final.
- Afectação dos postos e operários aos produtos.
- Critérios de nivelamento das ordens na célula final.
- Algoritmo de produção de componentes exóticos.
- Tamanho de lotes a produzir na célula de preparações.

Capítulo 5

Desenvolvimento do modelo de simulação

5.1 - Introdução

No capítulo quatro foi apresentado o sistema de produção, assim como todas as entidades intervenientes na produção. Neste capítulo irá ser apresentado o desenvolvimento do modelo. A implementação do modelo foi realizada com o recurso à ferramenta AnyLogic, na qual foi desenhado o modelo que representa o sistema em estudo e é constituído pelas diferentes entidades anteriormente apresentadas.

Neste estudo, a simulação foi feita utilizando-se o *software* de simulação AnyLogic Advanced versão 6.2.1, com a licença educacional disponibilizada pela FEUP.

5.2 - Entidades

Na tabela seguinte pode ser visto a correspondência entre as entidades e a implementação no AnyLogic.

Tabela 5.1 - Correspondência entre as entidades do modelo e da implementação

Modelo conceptual	Implementação
Buffer	*Queue*
Caixa de nivelamento	*Active Object*
Ordem	*Java Class*
Produto	*Active Object*
Componente	*Java Class*
Célula Final	*Active Object*
Supermercado	*Active Object*
Célula de Preparações	*Active Object*
Caixa	*Java Class*
Posto	*Active Object*
Milk run	*Agent*
Operário	*Agent*
Ordem Célula	*Collection Variable*

5.3 - Caixa de nivelamento

O *Active Object* caixa de nivelamento é constituído por quatro listas. Cada uma das listas contém as ordens de uma célula final. A sequência das ordens nas listas é igual à sequência de entrada, isto é, quando entra uma nova ordem na lista irá ocupar a última posição.

Existem também quatro funções *getOrdem* para cada célula final, esta função recebe como argumento uma variável do tipo inteiro, que será a posição da ordem na lista e retorna a ordem presente nessa posição.

As ordens são instâncias de uma java *class Ordem*, a qual tem apenas um atributo. Este atributo é a referência do produto, é único visto que a quantidade da ordem é normalizada e é sempre igual a dezasseis.

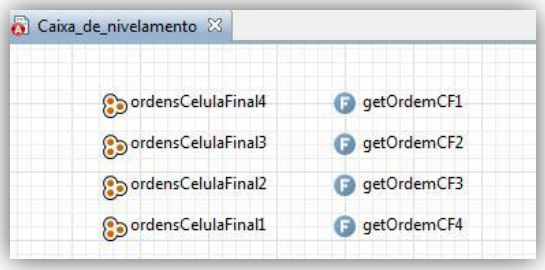

Figura 5.1 - *Active Object* Caixa de nivelamento.

5.4 - Produto

O Produto é mais um *Active Object* deste modelo. Neste *Active Object* encontram-se as listas de componentes de cada produto, logo para cada produto há uma lista correspondente. As listas podem ser identificadas na figura 5.2 com o nome *produtoref* seguido da referência do produto.

Os componentes são instâncias de uma java *class Componentes*, a qual tem como atributos a referência do componentes e a quantidade.

Outros constituintes do Produto são as funções *getProdutoActual* e *getProduto* para cada célula final. A função *getProdutoActual* quando chamada retorna os componentes do produto da ordem actual da célula final, a diferença da função *getProduto* é que esta retorna componentes da ordem recebida no argumento.

O parâmetro *caixa_de_nivelamento* faz a ligação entre este *Active Object* e *o Active Object Caixa_de_nivelamento*.

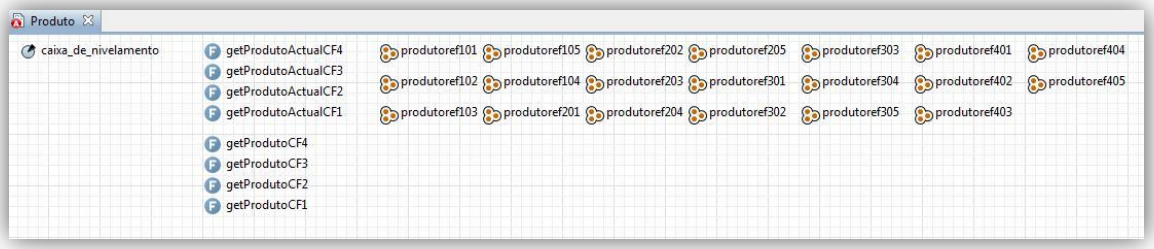

Figura 5.2 - *Active Object* Produto

5.5 - Célula final

A célula final é a responsável pela produção dos produtos, numa breve análise da figura 5.3 podemos identificar os diferentes constituintes do *Active Object* célula final, tais como funções, variáveis, listas, parâmetros, *queues*, enters, um *statechart* e um *selectOutput*.

Os parâmetros *célula*, *produto*, caixa_*de_nivelamento* e *milkrun* são utilizados para estabelecer a ligação da célula final aos *Active Objects* com o mesmo nome, o parâmetro parque tem como função informar *o milk run* onde parar quando se desloca para a célula final.

Quanto às variáveis, a *actual* contém o número da ordem actual a ser produzida, a variável *actualflag* é apenas uma auxiliar na programação. As variáveis *startrotura* e *totalrotura* permitem o cálculo da variável *taxaroutura* que indica a percentagem de tempo em que a célula final está em rotura e consequentemente parada por falta de caixas com os componentes necessários. Neste campo temos ainda a informação do número de roturas pela leitura da variável *numeroroturas*.

No que se refere a listas tem a lista de referências dos componentes que a célula final trabalha, e também a lista *emFalta*, é nesta lista que estão os componentes que faltam para a produção.

As *queues* são um objecto da *Enterprise Library* e representam os *buffers*. As *queues* ligadas ao objecto *selectOutput* são os *buffers* do bordo de linha, sendo que a *queue3* é o

buffer para os componentes com abastecimento do tipo caixa cheia/caixa vazia e exóticos, e os outros três para componentes com abastecimento por sequência. Exactamente em frente a cada um desses *buffers* encontram-se os *buffers* onde são colocadas as caixas vazias.

O objecto que representa a dinâmica interna da célula final é o *statechart*, na realidade os verdadeiros estados em que a célula final estará são o estado *inicio*, *faltaCaixa* e o *operação*, pois os estados *verificaExistenciaDeCaixa* e o estado *retiraCaixa*, são estados que apenas realizam a operação igual ao nome do estado correspondente.

Por fim temos as funções, as funções *verificaEmFalta* e *pickVaziaToMilkrun* são funções chamadas pelo *Milk run* quando se desloca até à célula final, no entanto a função *verificaEmFalta* chama outras funções internas e externas à célula final, tal como foi apresentado no algoritmo da função na figura 5.4 A função *pickVaziatoMilkrun* colocas as caixas vazias no *Milk run*. Os algoritmos das funções *verificaQueTemCaixa* e *pick* podem ser vistos nas figuras 5.5 e 5.6 respectivamente. As funções *bufferCheia*, *bufferVazia* e *enterVazia* são usadas apenas como auxiliares na correspondência das referências das caixas com o respectivo *buffer* ou *enter*.

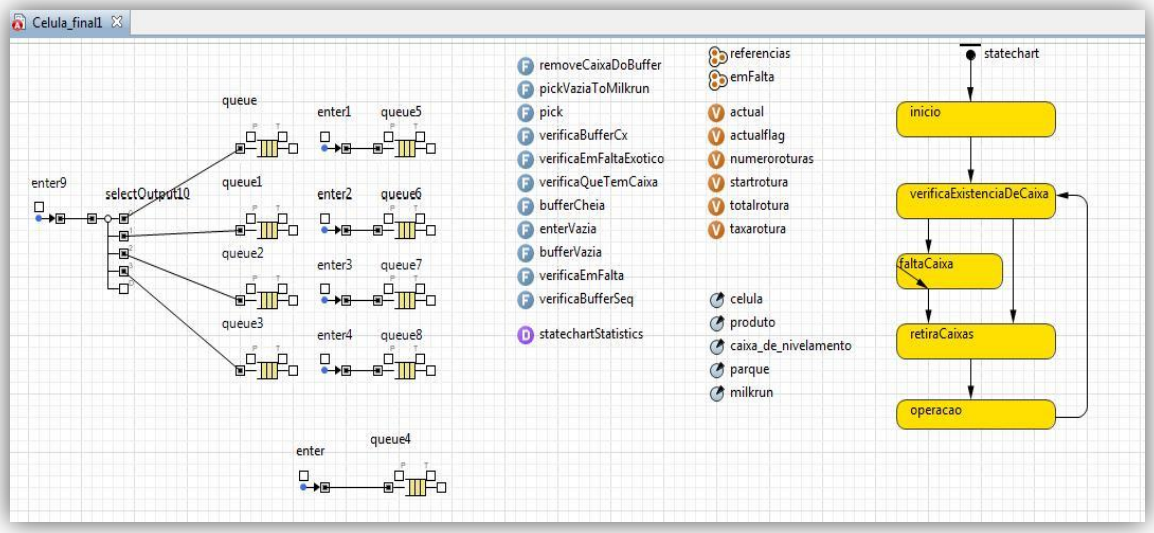

Figura 5.3 - *Active Object* Celula Final

Algoritmo da função *verificaEmFalta*()

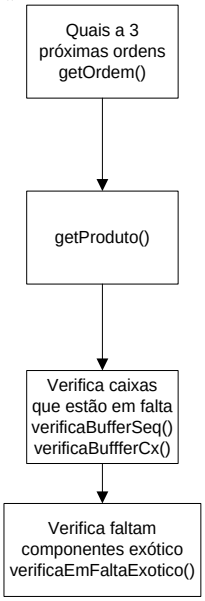

Figura 5.4 - Algoritmo da função verificaEmFalta()

Algoritmo da função *verificaQueTemCaixa*()

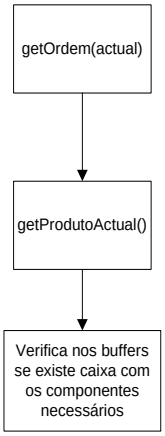

Figura 5.5 - Algoritmo da função verificaQueTemCaixa()

Algoritmo da função *pick*()

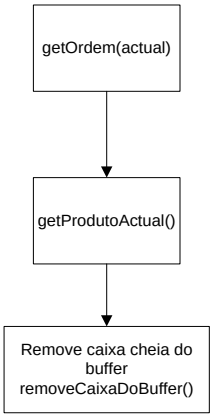

Figura 5.6 - Algoritmo da função pick()

5.6 - *Milk run*

O *milk run* tem como função abastecer a célula final a que está associado, fica então explicada a existência do parâmetro *celula_final*, que efectua a ligação. O mesmo sucede com os parâmetros célula e supermercado que estabelecem a ligação com a célula de preparações e o supermercado respectivamente.

A dinâmica desta entidade encontra-se modelada pelo *statechart*, pode-se verificar na figura 5.7 que os estados presentes no *statechart* estão directamente relacionados com o trajecto que o *milk run* efectua. A transição dos estados *moverParaCelulaFinal* e *moverParaSupermercado* é disparada quando o *milk run* chega ao *parque* associado, quer ao do supermercado quer ao da célula final.

No estado *chedadaACelulaFinal* o *milk run* interage com a célula final, neste momento chama a função *dropToCelulaFinal* que retira as caixas cheias para a célula final. Quanto ao estado *chegadaASupermercado*, são chamadas as *dropVaziasToSupermercado*, que retira as caixas vazias para o supermercado, e a função *pickCheiasFromSupermercado*, que retira as caixas cheias do supermercado. As restantes funções têm como objectivo o auxílio da interacção do *milk run* com o supermercado.

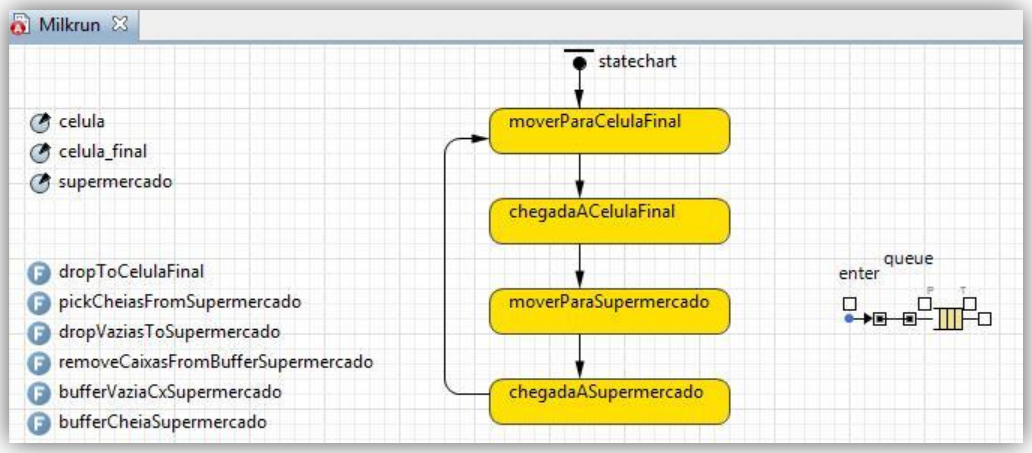

Figura 5.7 - ***Active Object* Milk run**

Algoritmo da função *removeCaixaDoBufferSupermercado*()

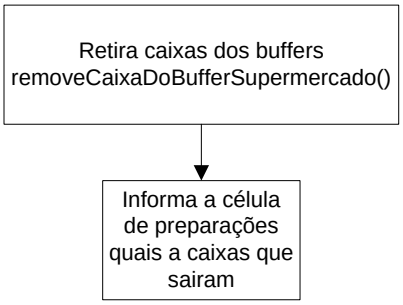

Figura 5.8 - Algoritmo da função *removeCaixaDoBufferSupermercado*

5.7 - Supermercado

O *Active Object* Supermercado é responsável pelo armazenamento de todas as caixas de componentes produzidas na célula de preparações. Para o armazenamento o supermercado tem um *buffer* associado a cada referência dos componentes. Na figura 5.9 estão representados os *buffers*, que são *queues* da *Enterprise* Library, os quais estão ligados a objectos do tipo *source*, este objecto é responsável pela criação das caixas de componentes, a criação das caixas é controlada na célula de preparações, ou seja quando a célula de preparações termina a produção envia uma ordem para a *source* correspondente para introduzir uma nova caixa no *buffer*. A função *sourceRef* é uma função auxiliar do método anteriormente descrito. Existe ainda um buffer dedicado à recepção das caixas vazias.

Quanto aos parâmetros *parque*, têm como função indicar a cada um dos *milk runs* qual o local a que se devem dirigir quando este pretende deslocar-se até ao supermercado. A variável *nivelsupermercado* indica qual o número de caixas inicial presente nos *buffers*.

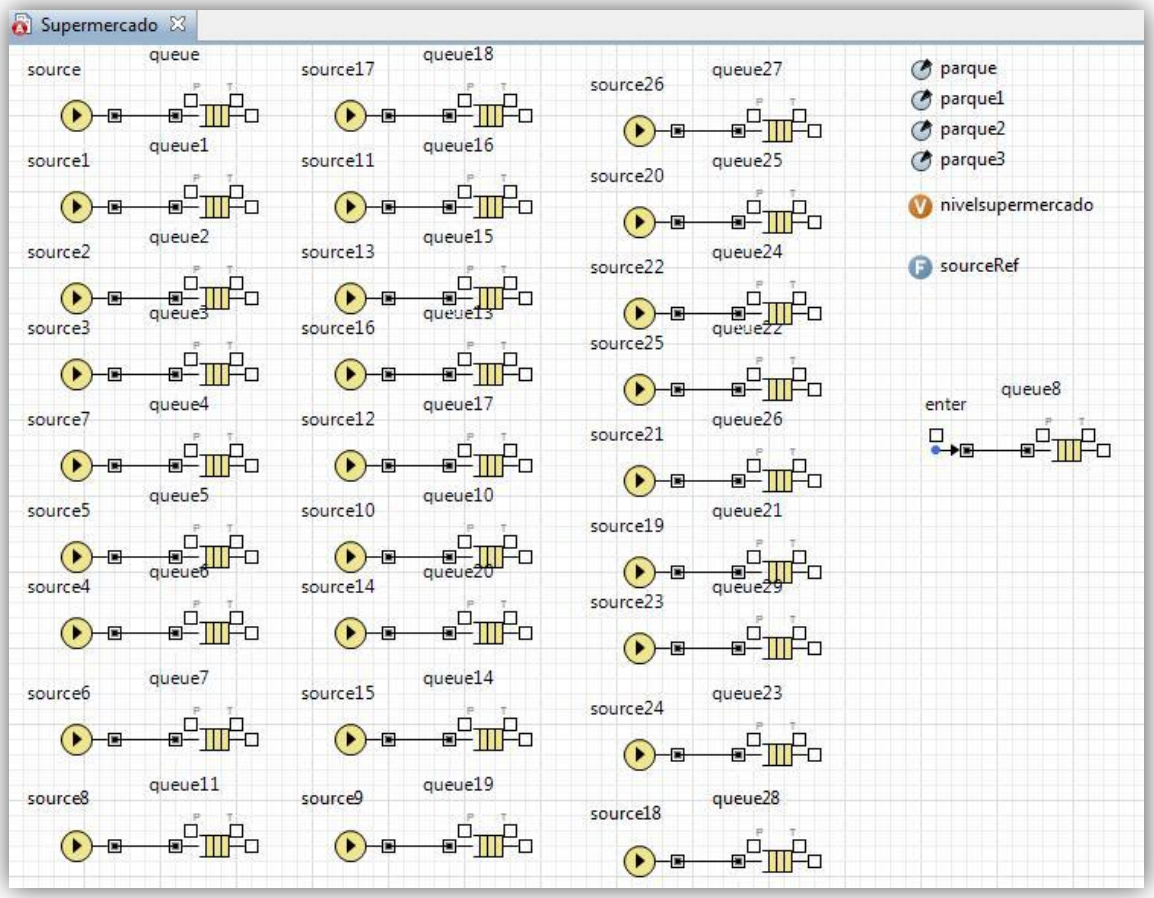

Figura 5.9 - *Active Object* Supermercado

5.8 - Célula de preparações

A célula final é composta por seis *Active Objects* posto e por três *Agents* operário que serão descritos mais à frente neste trabalho.

Por agora centrar-se-á atenções nas listas e nas funções. Verifica-se a presença de uma lista de postos e uma de operários contendo cada uma respectivamente os postos da célula de preparações e os operários. As listas *refposto* são compostas pelas referências de componentes que cada posto está habilitado a produzir, a lista *quantidadesproduzidas* tem o número de caixas produzidas de cada referência, e tem-se ainda a lista *caixasquesairam* que contém a informação das caixas que saíram do supermercado e por consequência as caixas que a célula de preparações terá de produzir.

A função *postoRef* é uma função auxiliar que identifica o posto no qual se pode produzir determinada referência. As funções *getOperarioLivre* e a *getPostoRefLivre* retornam um operário livre, e um posto livre para uma determinada referência, respectivamente. Por fim a função *tempoOperação* retorna o tempo necessário para produzir cada referência dos componentes.

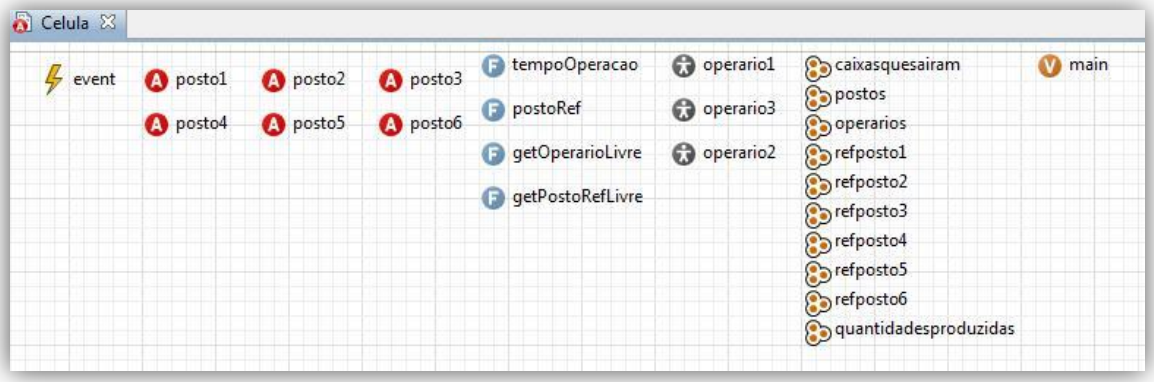

Figura 5.10 - *Active Object* Célula de preparações

A dinâmica da célula de preparações é representada pelo *Actionchart* da figura 5.11, pode-se observar que este tem início com uma consulta à lista *caixasquesairam*, de seguida verifica-se se existe um operário livre e também um posto que produza a referência da caixa e se está livre, caso todas as condições sejam verificadas procede-se à interligação do posto com o operário.

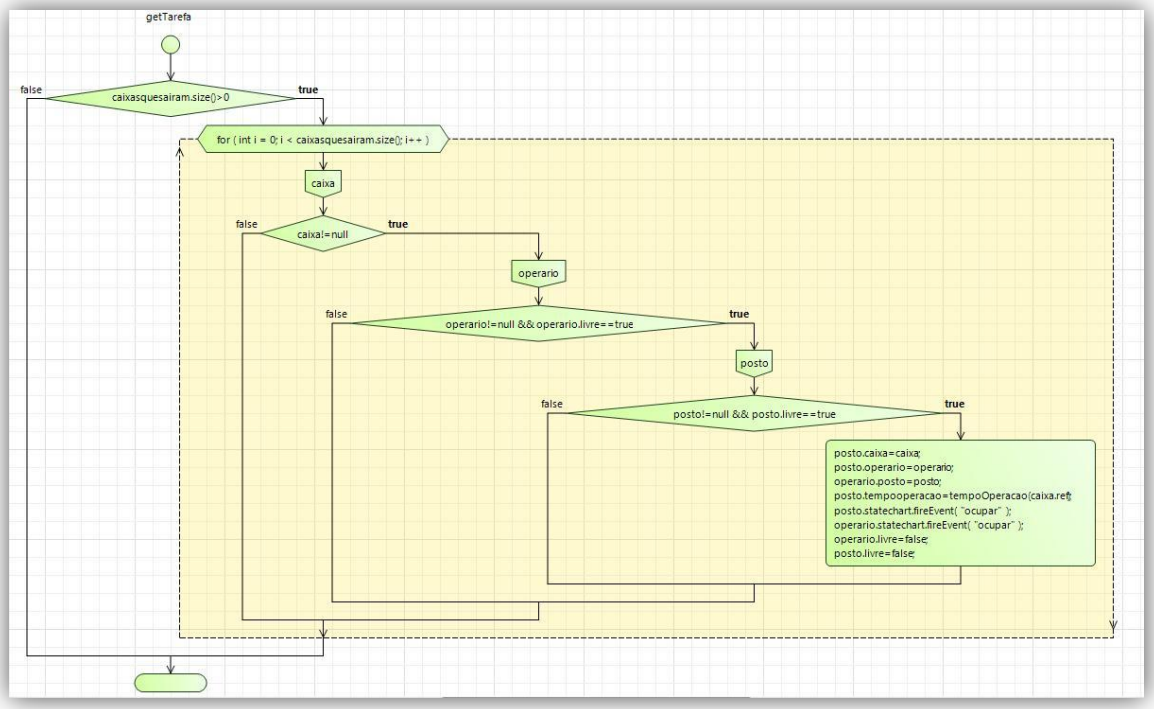

Figura 5.11 - *ActionChart* getTarefa

5.9 - Posto

O *Active Object* posto tem um *statechart* com dois estados possíveis, a passagem do estado *livre* para o estado *ocupado* dá-se quando o *statechart* recebe a mensagem *ocupar* enviada pela célula de preparações, recebe ainda os parâmetros caixa, operário, e *tempooperação* os quais são necessários para operação. A transição entre o estado ocupado e o estado livre é efectuado após o tempo de operação terminar.

O parâmetro parque define o local para o qual o operário deve deslocar-se. As variáveis *startWorking* e *totalWorking* auxiliam o cálculo da taxa de utilização do posto enquanto a variável *numerocaixasfeitas* conta o número de caixas produzidas.

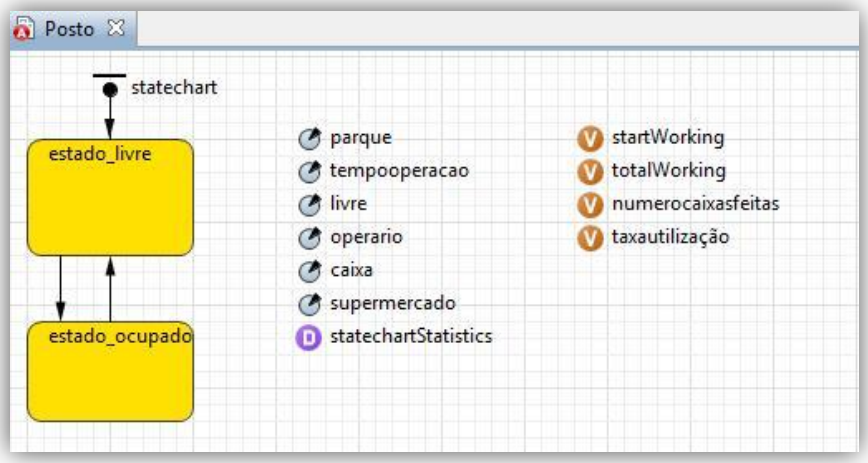

Figura 5.12 - *Active Object* Posto

5.10 - Operário

A implementação da entidade operário é semelhante à do posto. A passagem do estado *livre* para o estado *ocupado* tal como o posto é determinada pela recepção de uma mensagem *ocupar* enviada pela célula de preparações, mas a passagem do estado *ocupado* para o estado *livre* acontece com a recepção de uma mensagem *fim de operação* enviada pelo posto.

Tal como no posto as variáveis *startWorking* e *totalWorking* auxiliaram o cálculo da taxa de utilização do operário. E parâmetro posto estabelece a ligação com o posto e o parâmetro *parquelivre* determina o local onde deve permanecer o operário quando se encontra no estado livre.

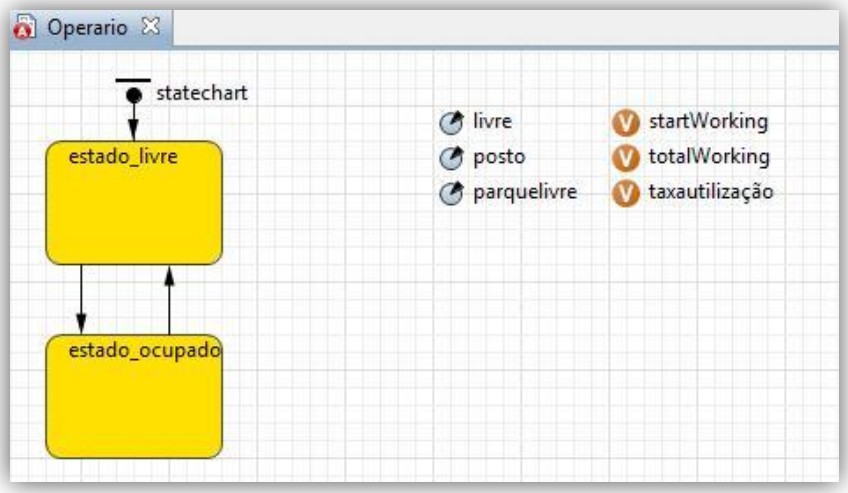

Figura 5.13 - *Active Object* Operário

5.11 - Main

Após apresentação de todos os *Active Objects* presentes no sistema, passar-se-á ao *Active Object Main* o qual é a raiz do nosso sistema, é aqui que surgem todos os elementos do sistema de produção, como pode ser visto na figura 5.14. Também neste elemento se faz a ligação com a base de dados que contém as ordens de produção da célula final. Neste caso a base de dados utilizada foi a Microsoft Office Access 2007. A base de dados é constituída por quatro tabelas, uma para cada célula final, sendo que cada tabela tem como campos o número de sequência e a referência do produto.

Ainda a referir que as caixas são instâncias da classe Caixa a qual tem como atributos a referencia e a quantidade, esta classe é ela própria uma extensão da *class entity* do Anylogic.

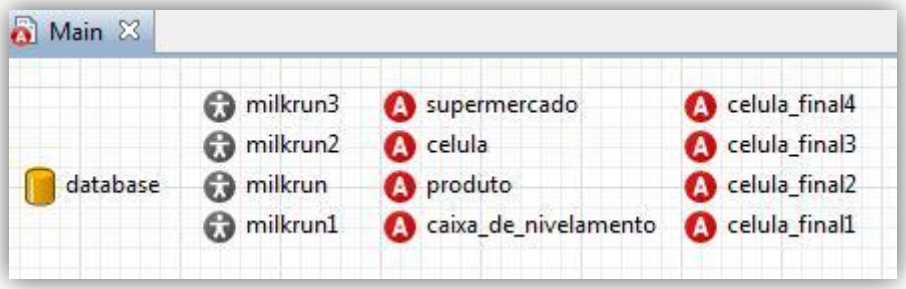

Figura 5.14 - *Active Object* Main

5.12 - Animação

A animação pretende mostrar, o fluxo e as actividades das entidades no sistema ao longo do tempo de execução do modelo. Esta representação visual do sistema permite a quem utiliza o simulador analisar informações importantes e a sua variação durante a execução.

Com o recurso à animação é possível um acompanhamento pormenorizado do funcionamento do sistema de produção em todos os momentos de execução.

Na animação geral do sistema de produção é facilitado o acesso a informações das células finais com o estado em que se encontram, o nível a que estão os buffers do bordo de linha, a ordem que estão actualmente a produzir, o número de roturas e a percentagem de tempo em rotura.

Nas figuras 5.15 e 5.16 estão representadas as animações, da célula de preparações e geral do sistema do sistema de produção respectivamente.

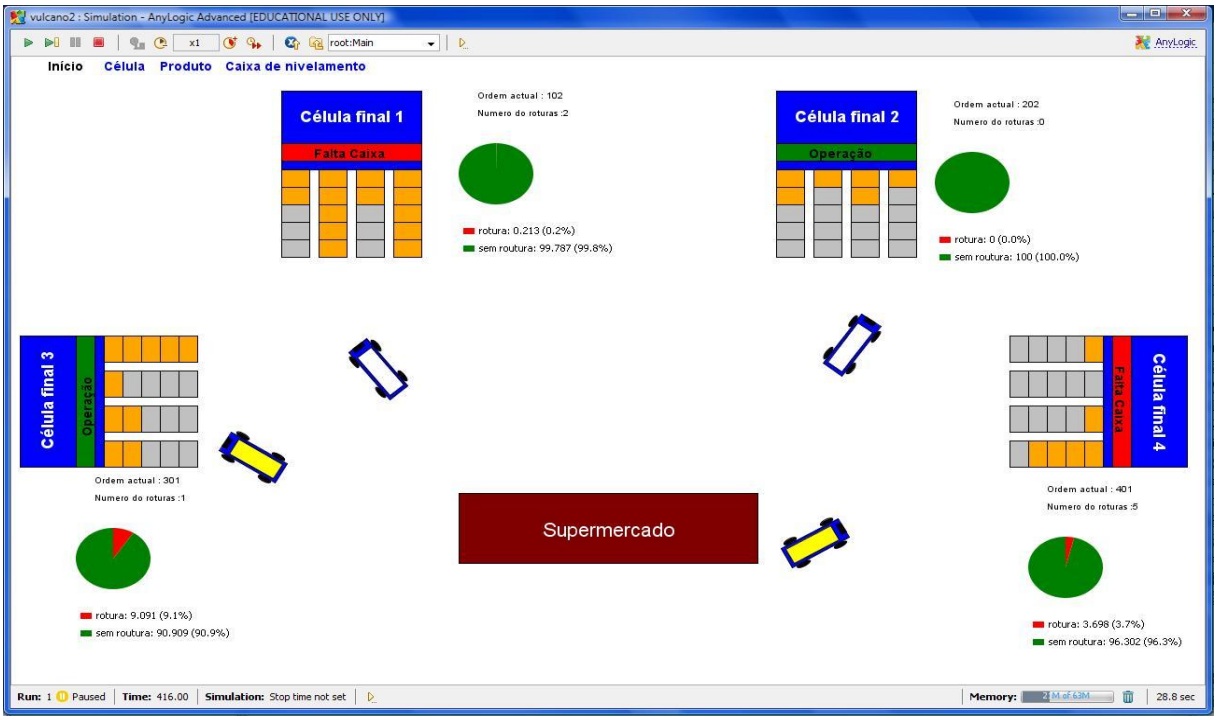

Figura 5.15 - Animação geral do simulador

Na animação da célula de preparações é oferecido o acesso a informação de diferentes aspectos:

- Número de caixas feitas de cada referência.
- Histograma da taxa de utilização dos postos.
- Histograma da taxa de utilização dos operários.

Postos

- Taxa de utilização.
- Número de caixas produzidas.
- Estado actual.
- Quais as referências que podem ser produzidas.

Operários

- Taxa de utilização.
- Estado actual.

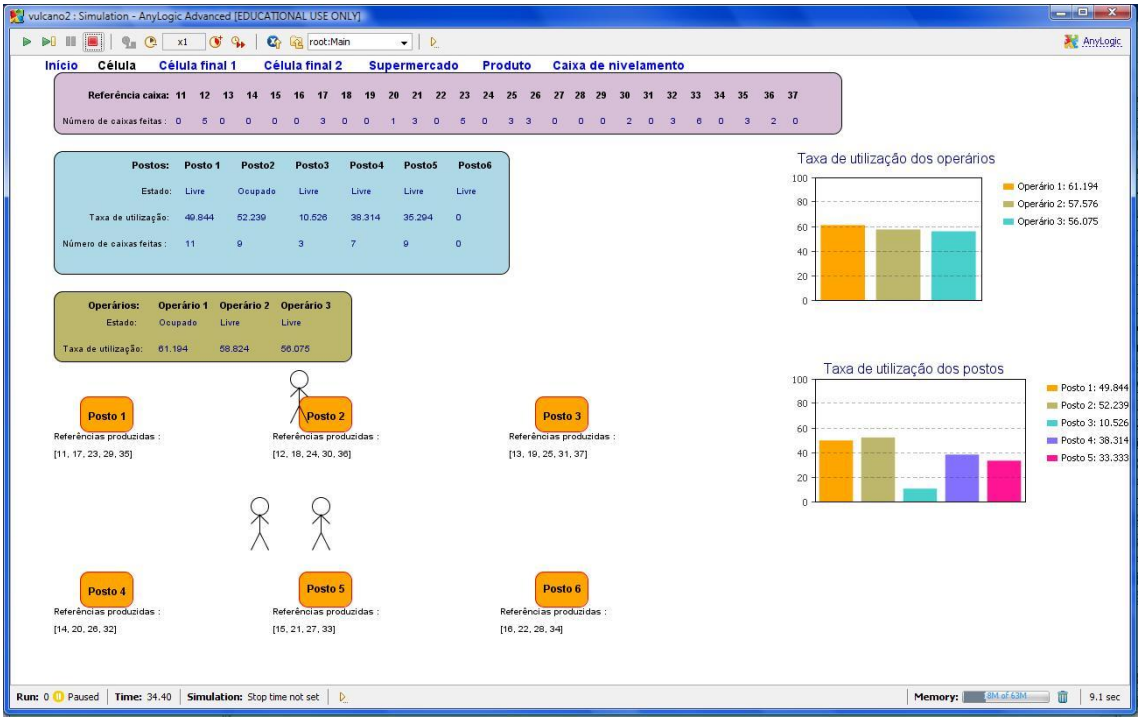

Figura 5.16 - Animação da célula de preparações

5.13 - Cenários de simulação

No capítulo 4 foram identificados vários aspectos do sistema que podem ser objecto de avaliação. Neste caso de estudo, optou-se por considerar os seguintes:

- Dimensionamento dos buffers do supermercado, ou seja a quantidade média de caixas que cada buffer do supermercado deverá conter.
- Número de operadores da célula final.

Foram considerados os seguintes cenários:

Cenário 1: três operários, nível dos buffers do supermercado igual a quatro.

Cenário 2: três operários, nível dos buffers do supermercado igual a três.

Cenário 3: três operários, nível dos buffers do supermercado igual a dois.

Cenário 4: dois operários, nível dos buffers do supermercado igual a dois.

Cenário 5: dois operários, nível dos buffers do supermercado igual a três.

Cenário 6: dois operários, nível dos buffers do supermercado igual a quatro.

A avaliação dos diferentes cenários foi baseada nos seguintes indicadores de desempenho: percentagem de roturas na célula final, taxa de ocupação dos operários, taxa de ocupação dos postos da célula de preparações, número de caixas produzidas nas células, número de caixas de cada referência produzidas nas células.

No ponto seguinte são apresentados os resultados obtidos para estes diferentes cenários.

Antes de executar o modelo, no entanto, foi preciso a sua verificação e validação.

No capítulo dois são referidas algumas técnicas para a verificação da simulação, uma das quais baseada na utilização do "*debuger*". No entanto, esta técnica não pôde ser utilizada

porque embora o AnyLogic integre um "*debuger*", esta funcionalidade apenas está disponível na sua versão profissional.

A verificação do modelo consistiu em:

- Testar valores de entrada: a alteração dos valores dos parâmetros de entrada na execução do modelo permitiu avaliar o comportamento do modelo e verificar como se comportavam os parâmetros de saída.
- Observar a animação: através da animação, houve a possibilidade de detecção de determinados erros relacionados com a interacção dos diferentes componentes do sistema através da visualização dinâmica, do comportamento do modelo desenvolvido, durante a execução do mesmo.

No que diz respeito a validação dos resultados, não foi realizada devido à escassez de tempo sendo que esta etapa requer bastante tempo visto ser necessária uma recolha exaustiva de dados do sistema real.

5.14 - Resultados

Os resultados obtidos estão sumariados nas tabelas seguintes:

Tabela 5.2 - Resultados obtidos para as células finais.

	Célula final 1		**Célula final 2**		**Célula final 3**		**Célula final 4**	
	Número de roturas	Percentagem de tempo em rotura	Número de roturas	Percentagem de tempo em rotura	Número de roturas	Percentagem de tempo em rotura	Número de roturas	Percentagem de tempo em rotura
Cenário1	2	0,2	1	0,1	6	1,1	0	0
Cenário2	2	0,2	1	0,1	5	2,7	0	0
Cenário3	2	2	1	1,9	5	5	0	0
Cenário4	4	3,8	1	1,3	5	6,3	3	1,3
Cenário5	3	2,6	1	2,1	7	5,6	0	0
Cenário6	2	0,1	1	0,1	7	3,8	0	0

Tabela 5.3- Resultados obtidos para os postos

	Posto1		**Posto2**		**Posto3**	
	Número de caixas produzidas	Taxa de utilização	Número de caixas produzidas	Taxa de utilização	Número de caixas produzidas	Taxa de utilização
Cenario1	133	35,8	158	44	129	43,8
Cenario2	131	35,5	156	43,8	126	43,2
Cenario3	128	34,62	154	43,5	124	42,5
Cenario4	122	32,9	146	41,1	124	42,5
Cenario5	126	34,1	148	42,1	121	41,1
Cenario6	129	34,8	153	43,1	126	43,2

Tabela 5.4- Resultados obtidos para os postos

	Posto4		Posto5		Posto6	
	Número de caixas produzidas	Taxa de utilização	Número de caixas produzidas	Taxa de utilização	Número de caixas produzidas	Taxa de utilização
Cenário1	76	22,7	95	18,3	133	31,6
Cenário2	75	22,5	95	18,3	131	31,1
Cenário3	74	21,8	95	18,3	129	30,5
Cenário4	77	22,9	95	18,3	124	29,3
Cenário5	72	21,2	92	17,9	128	30,4
Cenário6	72	21,73	93	17,9	129	30,5

Tabela 5.5- Resultados obtidos para os operários

	Operário1	Operário2	Operário3
	Taxa de Utilização	Taxa de Utilização	Taxa de Utilização
Cenário1	71,6	66,1	58,2
Cenário2	71	63,6	59,1
Cenário3	73,5	62,2	55,2
Cenário4	93,9	92,8	-
Cenário5	93,3	92,6	-
Cenário6	95,2	95	-

Com análise das tabelas de resultados é possível retirar as seguintes constatações:

- Utilização dos postos da célula de preparações manteve-se mais ou menos constante, assim como o número de caixas produzidas.
- O número de roturas nas células finais também não sofreram grandes alterações, o que era possível esperar visto que a produção na célula de preparações também não variou.
- Na taxa de utilização dos operários houve uma grande variação quando se passou de três operários para dois.

Uma análise global dos resultados obtidos através da simulação do sistema de produção permite concluir que é possível operar apenas com dois operários na célula de preparações e com um número médio de caixas no buffer do supermercado igual a três, sem que haja um aumento significativo na percentagem de tempo em rotura das células finais.

Capítulo 6

Conclusões e trabalho futuro

6.1 Satisfação dos objectivos

Conforme foi afirmado no primeiro capítulo, o objectivo principal deste projecto era conhecer a ferramenta de simulação AnyLogic e avaliá-la na perspectiva da simulação de sistemas de produção, em especial sistemas de produção *lean*.

No projecto a ferramenta AnyLogic foi aplicada na simulação de um sistema de produção o que permitiu experienciar a sua capacidade de modelação e avaliar as suas potencialidades na perspectiva da simulação de sistemas de produção.

O facto de reunir no mesmo ambiente de desenvolvimento as abordagens de sistemas a eventos discretos, a dinâmica de sistemas e sistemas baseados em agentes, conferiu uma grande flexibilidade na construção dos modelos.

A utilização de agentes revelou-se uma forma natural de modelação dos sistemas de produção *lean*, em que cada elemento do sistema (células, *milk runs*) tem um comportamento autónomo cuja modelação através de agentes é "natural".

O facto de a ferramenta incluir simulação a eventos discretos, nomeadamente *statecharts*, também se revelou muito útil pois em vários casos para modelar o comportamento interno das entidades envolvidas no sistema.

A possibilidade de representar algoritmos através de *actioncharts* também contribui para a legibilidade dos modelos.

Graças à *Enterprise Library* do Anylogic é possível modelar problemas, onde existem um fluxo de actividades bem delineado, com grande facilidade, pois os objectos existentes na biblioteca conseguem responder a grande parte das situações que possam ocorrer na realidade. E nos casos em que tal não acontece, a capacidade de criar objectos à medida, resolve essas situações. Também é importante realçar o facto de os objectos da *Enterprise Library*, facilitarem criar animações, com funções próprias para cada objecto, o que também permite separar o modelo interno da visualização.

Na realização deste trabalho, deparou-se com uma grande dificuldade decorrente da escassez de bibliografia específica, necessária para desenvolver a programação no Anylogic.

6.2 Trabalho futuro

Sendo o conhecimento da ferramenta AnyLogic, um dos objectivos, tal não pode ser atingido na sua totalidade apenas com um caso de estudo. Pois que seria necessário o estudo de diversos tipos de sistemas para avaliar em cada tipo qual o comportamento da ferramenta e profundidade que é capaz de alcançar.

Assim prevêem-se os seguintes desenvolvimentos futuros:

- No âmbito da avaliação o AnyLogic, o desenvolvimento de sistemas onde seja necessária a modelação dinâmica, pois neste estudo a mesma não foi necessária.
- No que diz respeito ao caso de estudo, um aprofundamente dos comportamentos dos agentes, tais como avarias, manutenção, pausas para almoço, etc.
- Ainda no caso de estudo a modelização de todo o sistema de produção, ou seja a inclusão de todas as células de preparações assim como o seu abastecimento.
- Utilização mais aprofundada da programação orientada objecto e na criação de entidades genéricas que possam ser instanciados (caso das células finais).

Referências

1. **Ehrlich, Pierre Jacques.** *Pesquisa operacional: curso introdutório.* São Paulo : Atlas, 1985.

2. **Schriber, T. J.** The Nature and Role of Simulation in the Design of Manufacturing Systems. J. Retti and K. E. Wichmann. *Simulation in CIM and Artificial Intellinge Techniques.* 1987.

3. **Shannon, Robert E.** *Introduction to the art and science of simulation.* 1998.

4. *Introduction to Simulation.* **Banks, Jerry.** 2000. Proceedings of Winter Simulation Conference.

5. **Hillier, F. S. and Lieberman, G. J.** *Introduction to Operations Research.* Boston : McGraw-Hill Higher Education, 2005.

6. *An Anthology on the Homology of Simulation with Artificial Intelligence.* **Doukidis, G.I.** 1987, Journal of the Operational Research Society 38. Nº 8.

7. **Law, A. M. and Kelton, W. D.** *Simulation Modeling e Analysis.* s.l. : McGraw-Hill, 1991.

8. *Introduction to Modeling and Simulation.* **Maria, Amu.** 1997. Proceedings of the Winter Simulation Conference.

9. *An Introduction to Simulation Modeling.* **Centeno, Martha A.** 1996. Proceedings of the 1996 Winter Simulation Conference.

10. *Introduction to Modeling and Simulation.* **Carson II, John S.** 2005. Proceedings of the Winter Simulation Conference.

11. *An Integrated Methodology for Manufacturing Systems Design Using Manual and Computer Simulation.* **Paquet, Victor and Lin, Li.** 2003. Human Factors and Ergonomics in Manufacturing. pp. 19-40.

12. **Ferreira, Luís Carlos Ramos Nunes Pinto.** *Geração Automática de Modelos de Simulação de uma Linha de Produção na Indústria Electrónica.* Departamento de Produção e Sistemas, Universidade do Minho. Braga : s.n., 2003.