Faculdade de Engenharia da Universidade do Porto

Controlador de fábrica – Automação da linha de produção

Francisco Parente up201606250

Lúcio Almeida up201609022

Nuno Minhoto up201604509

Pedro Monteiro up201503105

Rúben Lopes up201606821

Relatório do Trabalho Prático realizado no âmbito da unidade curricular

Informática Industrial do

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

12 de junho de 2020

Índice

Introdução

Este documento serve como relatório do projeto elaborado na disciplina de Informática Industrial, o qual teve como foco a projeção e implementação de um sistema de controlo de uma fábrica.

De forma a conseguir chegar ao objetivo de automatizar a linha de produção não nos limitamos a apenas um controlo de certas partes da fábrica de forma autónoma, mas sim a criação de uma arquitetura que possibilitou uma implementação em vários níveis e que manteve os vários blocos constituintes do controlo interligados entre si e com possibilidade de comunicarem.

Para isso recorremos a várias ferramentas: o software *Codesys*, o protocolo *Modbus* para a comunicação entre o *Codesys* e o simulador, o protocolo OPC-UA para a comunicação entre o MES e o PLC virtual, o protocolo UDP para realizar a ligação ao software de gestão de recursos (ERP), e ainda TCP/IP para as comunicações com a base de dados.

De forma a demonstrar melhor o nosso trabalho, numa primeira parte iremos abordar a estrutura do código e como o mesmo funciona e por fim a implementação. Iremos depois fazer uma avaliação baseada no nosso projeto inicial comparando-o com o resultado final, tendo em conta possíveis problemas que tenhamos sentido na sua implementação.

Reservamos ainda um espaço onde comparamos a nossa arquitetura com os padrões existentes.

Para terminar uma breve descrição do trabalho em equipa, a apresentação de resultados e as conclusões que tiramos deste trabalho.

Estrutura do código

Na estrutura implementada projeto, definimos que o MES iria-se responsabilizar por gerir as ordens sistema, os caminhos atribuídos a cada peça, a adição de peças, e comando direto sobre todos os tapetes da fábrica, para além de gerar informação útil para o operador ao atualizar estatísticas de interesse relativamente ao sistema na base de dados. O PLC responsabilizar-se-ia por executar os comandos dados pelo MES para cada tapete de forma segura, de modo a que se por exemplo o comando dado pelo MES para um tapete máquina fosse de executar uma operação processamento, o PLC trataria de toda a questão de executar esse comando na fábrica, e indicando de volta ao MES quando começa a executar o comando, e quando termina o comando

Para o MES, teremos no diagrama de classes final a classe peça que irá guardar um array de comandos e um contador de movimentos, para sabermos em que comando vai a peça. Peças pertencem à classe Tapete, que pode conter uma peça ou não. Tendo em conta o Tapete em que a peça se situa, o comando atual da peça irá ser passado para o tapete no PLC correspondente a esse tapete no MES, para começar essa ação no tapete correspondente no chão fábrica. Assim as peças quando têm um comando movimentação, são trocadas de Tapete em Tapete no MES, tendo em conta a posição atual no chão fábrica, permitindo ao MES assim saber sempre em que tapetes se situam todas as peças presentes. A classe Fábrica irá conter todas estas classes, tendo um array de 47 Tapete, um para cada tapete presente no chão fábrica. A classe Fábrica vai ter as variáveis a ser partilhadas para o PLC, indo buscar estes valores todos aos tapetes e peças presentes no MES.

Teremos depois a fila_de_ordens e ordens_completas, que serão arrays da classe Ordem, que guarda toda a informação relativa a uma ordem no sistema e guarda uma classe Máquinas que gere informação relativa ao caminho que impõe nas peças que são adicionadas por esta ordem. Toda a gestão ordens baseia-se em cada ordem poder reservar máquinas/tapetes descarga chão fábrica para essa ordem, sendo que enquanto a reserva se mantém nenhuma outra ordem poderá usar essas máquinas. Quando termina de processar/descarregar todas as peças dessa ordem, liberta a reserva. Para este sistema de reservas, foi necessário as estruturas lista_máquinas e lista_pushers que guardam o valor de reservas para as máquinas e os tapetes descarga, respetivamente. A classe UDPcomm contém as funções necessárias para receber ordens e adicionar as ordens na fila_de_ordens e conseguir enviar informação relativa ao pedido request_stores. A classe GestorOrdens terá as funções necessárias para a gestão ordens e adição peças das ordens. A classe GestorPeças terá funções necessárias para que cada ordem consiga adicionar a peça da ordem e terá funções para decidir o caminho atribuído a cada ordem, para que essa ordem depois ao adicionar peça meta esse caminho na peça. A classe GestorTapetes terá função para percorrer todas as peças e atualizar os comandos da fábrica

tendo em conta vários parâmetros, e terá a função para inicializar a fábrica com os valores guardados na base dados e PLC. A classe DBcomm terá as funções para conetar a base dados, para meter na fábrica valores ordens e estatística guardados na base dados, e para atualizar os valores base dados com os valores ordens e estatística atuais na fábrica, para permitir a retoma atividade caso o MES se desligue. A classe OPCcomm terá as funções necessárias para permitir comunicação entre o MES e o PLC. A classe ConsolaFabrica é um programa adicional que permite aceder à estatística e dados da fábrica presentes na base dados, além de permitir fazer reset dos dados na base dados para quando se quiser recomeçar a fábrica de novo.

Em relação às classes associadas ao PLC, no diagrama apenas estão representadas as variáveis que partilha com o MES, pois todas as outras variáveis presentes servem para fazer a ligação entre os tapetes no PLC e para comunicação com o chão fábrica (por exemplo, todos têm uma variável sensor para indicar o valor do sensor tapete associado a esse tapete, também têm uma variável para indicar ao tapete seguinte que lhes vão passar uma peça, entre outros). A classe Máquina a, b ou c engloba já as classes Máquina a, Máquina b e Máquina c usadas no PLC, partilham as mesmas variáveis e atuam da mesma forma em relação aos comandos recebidos, mas fazem o processamento peças em tempos diferentes tendo em conta se são máquina a, b ou c, o tipo peça lá presente e a ferramenta atual na máquina, tudo valores que têm de ser passados do MES. A classe pecas guarda strings com a informação sobre as peças presentes atualmente no chão fábrica, pois como a base dados apenas guarda as ordens e estatística, guardando a informação das peças presentes no chão fábrica permite que caso o MES se desligue, permite retomar a operação destas peças. Se não, teria que se retirar todas as peças do chão fábrica para poder retomar as ordens presentes na base dados. Para todas as classes, têm o mesmo tipo variáveis, mas atuam de forma diferente para comandos diferentes tendo em conta o tipo de tapete que são. O programa PLC junta todas estas classes num único programa e faz as ligações necessárias entre eles.

Funcionamento do código

A descrição do funcionamento do código está assente nos diagramas de sequência anexos sobre o funcionamento da função main e das threads responsáveis pela conexão e troca de informação com a base de dados e pela comunicação UDP com o ERP.

Deste modo, no início do programa têm lugar a inicialização das variáveis, e o estabelecimento das ligações ao PLC e à base de dados de onde são lidos os valores guardados caso se trate de uma situação de falha do MES. Após isso ocorre a inicialização de todas as variáveis relativas à fábrica e são iniciadas as threads mencionadas anteriormente e o programa entra no loop infinito estando agora à espera da chegada de ordens.

A thread responsável pela comunicação UDP cria um socket para o efeito e fica bloqueada à espera de receber informação nesse mesmo socket. Quando é recebido o ficheiro XML vindo do ERP, ocorre o *parcing* deste e são criados os objetos da classe *ordem* com a mesma informação das ordens que foram lidas do ficheiro XML e estes objetos são introduzidos na *fila_de_ordens* onde estão armazenadas todas as ordens que se encontrem por processar, ou em processamento no MES. Após as ordens terem sido introduzidas na fila, a thread volta ao estado de espera por um novo packet vindo do ERP.

Entretanto no nosso main, começam por ser lidos os valores da fábrica através da conexão OPC-UA. Após isso é feita a operação de identificar as ordens. Caso seja uma ordem de transformação são identificadas as operações que a peça vai sofrer, identificando quais as máquinas, ferramentas e tempos a usar, e atribuir uma reserva de máquinas a essa ordem e o caminho que as peças devem seguir. Caso não seja possível reservar nenhuma máquina por ocupação das mesmas, as ordens são deixadas sem reserva e ficam em espera até haver recursos para a efetuar. Caso seja uma ordem de descarga, é verificada a disponibilidade dos *pushers* de descarga, e são atribuídos às ordens caso os mesmo se encontrem livres. Após as operações de reserva, caso haja ordens no sistema, a saída de peças do armazém esteja livre e haja peças no armazém para satisfazer as necessidades da ordem a executar, são enviadas peças das ordens que já possuem reservas de máquinas e caminhos determinados para as suas peças.

Seguido do envio de peças temos um mecanismo de deteção da chegada de peças aos seus destinos que permite acompanhar o percurso da peça e determinar quando é possível enviar uma nova peça da mesma ordem sem sobrecarregar a fábrica. No caso das operações de descarga, é acompanhado o número de peças que se encontram nos tapetes de descarga dos pushers, de maneira a não causar bloqueios por excesso de peças.

Após isso, são determinados quais os valores que vão ser dados para cada tapete da fábrica e esses comandos são enviados por OPC-UA para o PLC para este os executar.

Entretanto a thread da base de dados vai enviando os valores cruciais que é necessário guardar na base de dados de meio em meio segundo.

Implementação

Na implementação do MES, escolhemos a utilizar a linguagem de programação Java, por proposta de um dos membros do grupo, visto que esta linguagem é orientada a objetos, e alguns elementos do grupo já tinham experiência em trabalhar com a mesma. Para armazenar e organizar as ordens dentro do MES tiramos partido da estrutura de dados Array_list que nos permitiu guardar uma lista dos vários objetos da classe ordem.

Para a realização da comunicação OPC-UA com o plc utilizamos a biblioteca prosysopc.ua, que nos forneceu as funcionalidades necessárias para implementar essa ligação. Além das bibliotecas fornecidas por esta biblioteca, criamos alguns métodos na classe OPCcomm, que tirando partido das funções existentes na biblioteca realizava as diferentes funcionalidades que desejávamos, como iniciar conexão, ler valores e escrever valores.

Para a receção dos ficheiros XML provenientes do ERP, criamos um socket UDP que recebia o ficheiro e posteriormente era feita a sua análise com um método de parsing DOM (Document Object Model). Escolhemos este método por a dimensão dos ficheiros XML recebidos era pequena e com um número reduzido de elementos e este método permite a criação de uma árvore com nós e elementos que, devido ao tamanho reduzido de informação no ficheiro, permite aceder com facilidade a qualquer informação vinda do ficheiro.

De maneira a garantir a persistência da informação do MES utilizamos uma base de dados local, em que utilizamos as bibliotecas e funções fornecidas pela linguagem Java.

Face à necessidade de ter várias tarefas a executar em paralelo como aguardar e receber novas ordens UDP que cheguem, guardar a informação na base de dados e a gestão da informação dentro do MES decidimos usar threads para permitir todas estas tarefas a serem executadas ao mesmo tempo. Assim a classe aplicação contém o main() do programa, onde são inicializadas as variáveis e são criadas as threads, uma para receber as ordens vindas do ERP e outra para atualização dos valores na base de dados. Após todas as inicializações o main() entra num loop infinito onde são chamadas funções de outras classes responsáveis por organizar ordens, enviar novas peças para a fábrica e atualizar os valores e estado que cada elemento da fábrica tem.

No que diz respeito ao PLC utilizamos o codesys, uma vez que este consegue comunicar com o MES através de OPC-UA e através da comunicação ModbusTCP comunica com o simulador da fábrica. No primeiro caso o PLC funciona como servidor de OPC-UA fornecendo as variáveis para o MES alterar, enquanto que na comunicação com o simulador, o PLC assume a função de escravo na qual atualiza as variáveis Modbus periodicamente.

Estrutura final vs inicial

No início do semestre tínhamos uma vaga ideia de como iríamos abordar todos os aspectos deste trabalho. Por esse mesmo motivo, os nosso primeiros diagramas estavam um pouco incompletos. À medida que fomos avançando no projecto começamos a perceber que certas partes da nossa ideia inicial tinham de ser alteradas e, em alguns casos, até foi necessário acrescentar novos elementos para prosseguir com a execução do trabalho.

Além de alterar um pouco a estrutura do nosso antigo diagrama de Sequências também acrescentamos o GestorPeças, DBcomm, FILA_ORDENS, LISTA_MAQ, ORDENS_C e LISTA_PUSHERS.

Sentimos a necessidade de acompanhar o trajecto das peças na fábrica daí termos alterado um pouco o GestorOrdens. O UDP foi alterado para uma Thread à parte porque precisava de estar em constante observação.

Para atualizarmos os valores dos comandos a dar à fábrica foi necessário alterar o GestorTapetes para enviar e receber updates em tempo real da disponibilidade dos tapetes na fábrica.

Inicialmente tínhamos pensado em ter trajetos pré-definidos para cada tipo de peça mas vimos que isso não iria resultado. Portanto, como os trajetos podem variar mediante o estado da fábrica, criámos o GestorPeça para decidir o caminho que a peça irá percorrer até ao seu destino.

As nossas ideias iniciais para a DB eram muito vagas pois não sabíamos que funções iríamos precisar daí só termos o GestorDB no primeiro diagrama. Criamos uma Thread à parte para a DB para termos uma ligação constante.

No nosso Class Diagram foram adicionadas várias funções às classes já existentes porque, à medida que o projecto foi avançando, sentimos a necessidade de ter estas funções disponíveis.

Foram adicionadas várias novas classes:

Foram criadas as classes do PLC (que não constavam no primeiro diagrama) para controlar as máquinas, tapetes, pushers e o percurso da peça na fábrica.

A classe da consola fábrica foi acrescentada porque era necessário acedermos à informação contida na DB.

Era preciso saber que transformações eram necessárias para transformar a peça inicial na peça final e verificar a disponibilidade das máquinas necessárias para as mesmas, daí termos criado a classe Máquinas.

Algumas das mudanças já foram referidas no Diagrama de Sequência e estão lá representadas.

Comparação com Padrões Existentes

Analisando os padrões existentes, nomeadamente a norma ISA-95, e comparando-os com a arquitetura proposta pelo nosso grupo, podemos dizer que:

A nível hierárquico o nosso projeto encontra-se de acordo com os ditames da norma. No nível 3 está o MES responsável pelo controlo. O sistema que foi automatizado ocupa o nível 2 desta hierarquia.

Relativamente ao modelo de controlo proposto[1], é possível encontrar uma conformidade entre a norma e o nosso trabalho. No que toca às 12 Functions padrão:

Modelo de controlo - Functions	**Equivalente no projeto**
Production Control	GestorOrdens, GestorTapete, GestorPeca
Production Scheduling	GestorOrdens, GestorPecas
Product Inventory Control	ConsolaFabrica, DBcomm
Material and Energy Control	ConsolaFabrica
Order Processing	UDPcomm

No que toca às 31 information flows, várias estão presentes no nosso trabalho:

- *Schedule, Production from plan, Production capability:*
  - A nossa solução permite ler o estado da planta, que ações estão a decorrer e organizar as próximas ações baseado nas máquinas livres
- *Incoming order confirmation, Material and energy order requirements, Standards and customer requirements, Product and process requirements, Product and process information request, Production order, Release to ship:*
  - O nosso projeto possui vários métodos de forma a receber os parâmetros e requisitos para cada transformação e definir as transformações necessárias para chegar ao produto pretendido. É capaz de receber ordens e ainda mantém um registo das ordens e do estado do processo
- *Process data, Finished goods waiver, In-process waiver request, Finished goods inventory, Material and energy inventory*

[1] ISA-95 – " Functional Enterprise-Control Mode l"

- No nosso trabalho é mantido um inventário das peças existentes, das transformações realizadas, das possíveis falhas e de quantas peças ainda estão a ser processadas

Fazendo agora uma análise comparativa com o artigo "*Service Granularity in Industrial Automation and Control Systems"* salientam-se três abordagens, às quais é possível fazer uma correspondência no nosso trabalho.

A primeira, *Process Ganularity* apresenta um modelo muito semelhante ao implementado no MES. Onde temos vários serviços a controlar "major processes", mais concretamente os *GestorOrdens*, *ConsolaFabrica* e as threads responsáveis pela atualização da DB e pela comunicação através do envio de ficheiros XML. O objetivo da divisão foi não ter apenas um sistema centralizado, mas vários sistemas mais pequenos responsáveis e independentes pelas várias tarefas. Contudo não foi possível remover por completo a comunicação inter-serviços, mas foi muito atenuada.

A segunda, *Machine Granularity* é uma abordagem que em muitos aspetos se torna próxima daquela que está presente no sistema que nos propusemos a automatizar. Pelo artigo esta abordagem é descrita como mais detalhada a nível de monotorização e diagnóstico que lhe confere mais robustez, mais flexibilidade e menos custos de manutenção. Assim, a planta pode ser vista como uma só máquina, ou dividida nos seus vários componentes – tal como referido no exemplo do artigo. A planta é responsável executar um conjunto de tarefas à peça que lhe chega – transportar, transformar, encaminhar para destino. Uma vez recebidas as ordens de tratamento da peça depois a planta é responsável pela execução das mesmas. Algumas diferenças entre a abordagem do MES - *Process Ganularity* – esta abordagem *Machine Granularity* apresenta maior comunicação inter-serviços. A maior flexibilidade nesta abordagem é também um requisito presente na nossa fábrica. Se fosse desejado acrescentar mais um elemento – seja um tapete normal, rotativo, uma máquina ou um pusher – o número de serviços não sofria alterações, e os processos a executar, como são partilhados por todos os elementos com funções iguais seria uma adaptação fácil para o novo elemento incorporado.

Por fim, a arquitetura proposta em RAMI-4.0, esta referência apresenta três eixos gerais. *Hierarquia*: A nível hierárquico, os elementos estão ligados entre si. São divididos em três grandes níveis: Connected World, Smart Factory e Smart Products. Pressupõe que as várias células de produção trabalham de forma flexível e descentralizada. É ainda esperado que o produto (ativo na linha de produção) pudesse enviar informações e tomar decisões (no nosso caso seria a escolha

de caminho e máquina) – Smart Products.No nosso trabalho não há uma descentralização tão acentuada ao nível da fábrica nem uma independia tão grande ao nível do produto – peça. O produto, no nosso projeto, não executa nenhum papel ativo.

O segundo eixo, *Arquitetura*, está ligado com a verticalização da informação, as várias interfaces de comunicação, a interpelação e o uso da informação. Neste ponto, no nosso caso as comunicações entre o MÊS e a fárica recaíram sobre o protocolo OPC-UA, entre o MÊS e o software de gestão de recurso realizou-se com o protocolo UDP, a comunicação com a DB foi realizada por TCP/IP e por fim, o acesso aos sinais lógicos da fábrica (sensores e atuadores) foram enviados através do protocolo ModBus/TCP.

O último eixo que compõe esta referência é o *Ciclo de Vida.* No trabalho proposto, dentro deste tópico só foi abordado o ponto da *Produção.* Neste ponto estão contemplados aspectos como o produto, os dados e lote. Ao contrário de uma típica fábrica o nosso produto – a peça – não foi criado apenas transformado.

Organização da Equipa

Numa primeira fase, a de planeamento do projeto. Procedemos à análise dos requisitos exigidos para este projeto. Após o levantamento das tarefas procedemos à atribuição quer individualmente quer em par das mesmas.

De referir que a arquitetura inicial e o relatório final foi realizados em conjunto por todos os membros do grupo. Distribuição de tarefas:

- PLC - gestor de tapetes - Pedro
- Gestor de ordens- Nuno, Lúcio
- Data base- Rúben
- UDPCOM/XML-francisco
- Testes em Rede – Pedro, Nuno, Rúben
- Relatório – Lúcio, Francisco, Pedro, Nuno e Rúben

Ao longo do trabalho houve uma grande entreajuda, sendo que apesar da distribuição de tarefas não ficamos confinados cada um ao seu espaço. Sempre que necessário os membros do grupos estavam disponíveis para ajudar noutras tarefas que não a sua. Houve tarefas mais difíceis que outras contudo ninguém ficou isolado no seu trabalho. Desta forma propomos a seguinte distribuição da avaliação do grupo:

- Francisco – 20%
- Lúcio – 20%
- Nuno – 20%
- Pedro – 20%
- Rúben – 20%

Teste e resultados

Um dos testes efetuados na nossa fábrica consistiu simplesmente em iniciar o programa e enviar as ordens usadas na competição A e verificar os resultados obtidos:

Ordem	Nº Peças	Hora entrada	Prazo entrega	Hora fim ordem
P4->P8	7	18:12:11	18:17:11	18:15:23
P2->P6	30	18:12:05	18:17:05	18:16:48
P3->P5	7	18:12:08	18:17:08	18:17:19
P7->P9	10	18:12:08	18:17:08	18:18:14
P4->P5	6	18:13:33	18:30:13	18:18:33
P1->P9	20	18:13:33	18:28:33	18:24:59
descarga P9	8	18:12:11	-	18:15:43
descarga P6	6	18:12:11	-	18:15:21

Verificamos neste teste que a ordem P3->P5 e P7->P9 não conseguiu cumprir no tempo necessário. Em todas as outras ordens cumpriu o prazo entrega estabelecido.

Noutro teste, decidimos iniciar o programa e mandar as ordens de competição A, juntamente com uma ordem transformação 30 peças P4->P7, transformação que no nosso caso exige que uma peça vá da máquina C para máquina A (inclui transformação P8 em P7 na máquina A, que era pedido para os grupos 5 elementos), duas ordens descarga, uma de 7 peças P3 no tapete descarga 1 e outra 10 peças P5 no tapete descarga 3, durante a execução adicionou-se 5 peças carga P1, 5 peças carga P2 e desligou-se o MES duas vezes durante a execução do programa, para testar se ao se ligar novamente conseguiria retomar do ponto onde tinha sido desligado. Neste teste apenas se avaliou se a fábrica conseguiria processar as ordens todas pedidas, mesmo sendo desligada e ligada durante a execução. Passou neste teste, mas de referir que em alguns outros testes que se desligou e ligou o MES durante a execução, o MES não conseguiu retomar operação sem ocorrer algum erro no processamento peças, não se conseguindo fiabilidade total na retoma de operação das peças já existentes no chão fábrica.

Noutro teste, este que foi executado na fábrica logo após o teste anterior, quisemos testar o comportamento do programa quando a fábrica não tem peças suficientes no armazém para

continuar a executar a ordem. Para isto, decidiu-se adicionar duas ordens transformação, uma de 45 peças P2->P6 e outra de 20 peças P4->P5, sendo que quando foram adicionadas estas ordens o armazém da fábrica não tinha o número suficiente peças para conseguir fazer a ordem. Verificamos que o programa começou e processou as peças das ordens, até não haver mais peças disponíveis para as ordens. Quando ficou tudo parado, por não ter peças disponíveis, adicionamos uma ordem 15 peças P3->P4 para aumentar o número peças P4 para o armazém, e fomos adicionando peças de carga P2 à fábrica. Verificamos que, quando voltou a ter essas peças presentes no armazém, retomou o processamento da ordem, pois já tinha peças disponíveis, passando no teste.

Conclusão

De uma forma geral podemos dizer que este projeto foi concluído com sucesso.

A maior dificuldade sentida esteve no nível de otimização, sendo possível alcançar resultados melhores para este projeto. Apesar de não ser ideal, tentamos da melhor forma, dentro dos nossos conhecimentos chegar à solução ótima e, chegados agora ao fim, é nos possível compreender outras abordagens que teriam sido vantajosas para a implementação. Achamos que seria muito benéfico ao longo do MIEEC haver uma maior dedicação a este tópico – otimização de processos - no plano de estudo.

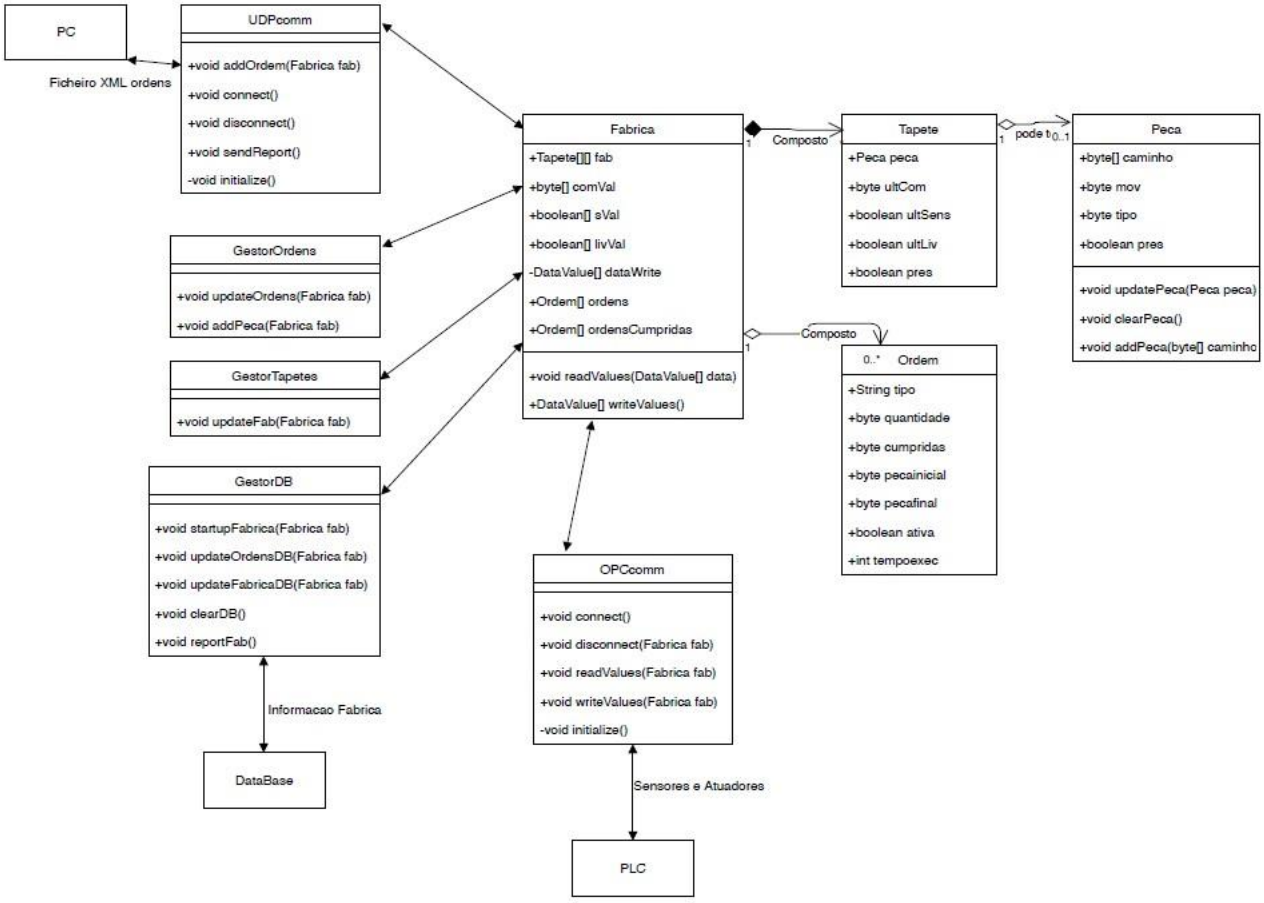
Contudo, conseguimos um sistema robusto e flexível, sendo possível introduzir elementos e alterar de definições através de um ficheiro (evitando uma reformulação geral do código).

Por outro lado, este trabalho possibilitou-nos um contacto mais real com o que se passa no mundo profissional, pois adquirimos conhecimentos técnicos ao nível da automatização de uma linha de produção, nomeadamente comunicações, implementação de um controlo por software, gestão de ordens a executar e gestão de recursos. Tudo isto tendo por base as referências e normas utilizadas na indústria real.

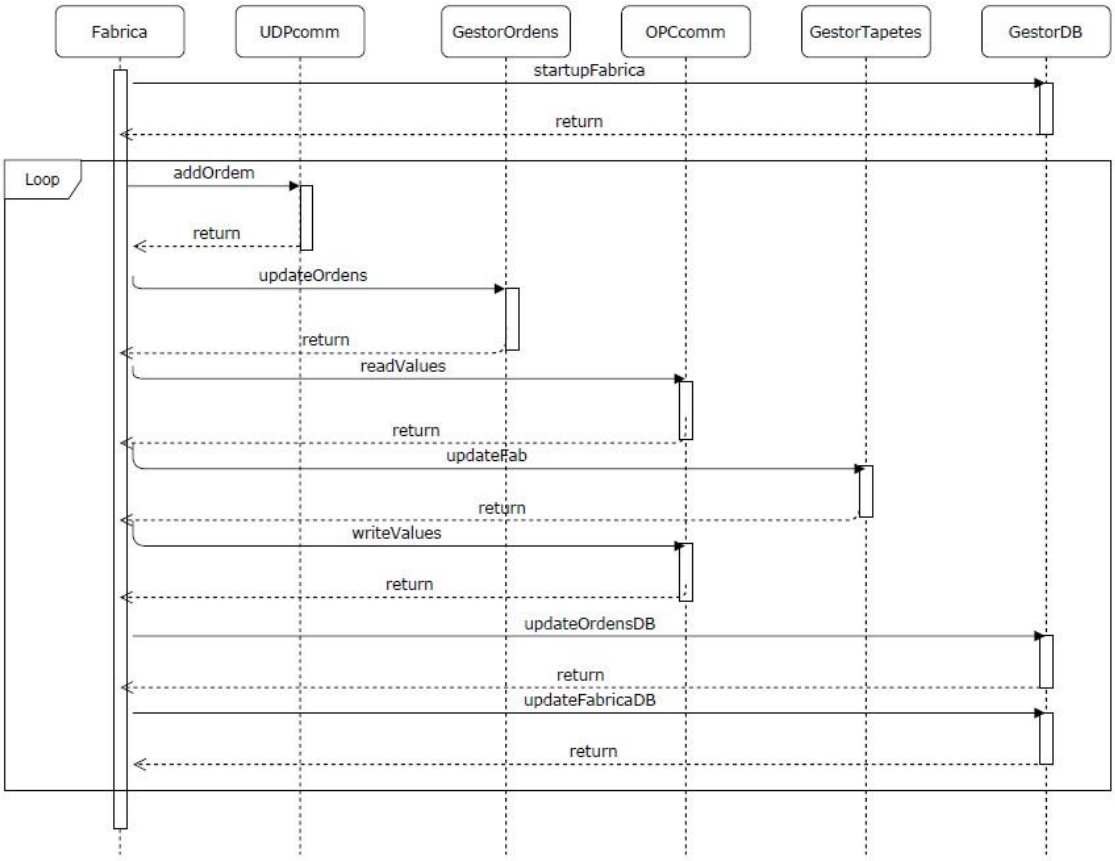
Concluindo, estamos satisfeitos com os resultados obtido e com todo o projeto implementado. Foi um desafio muito bom e certamente nos irá beneficiar no nosso futuro profissional.

Anexos

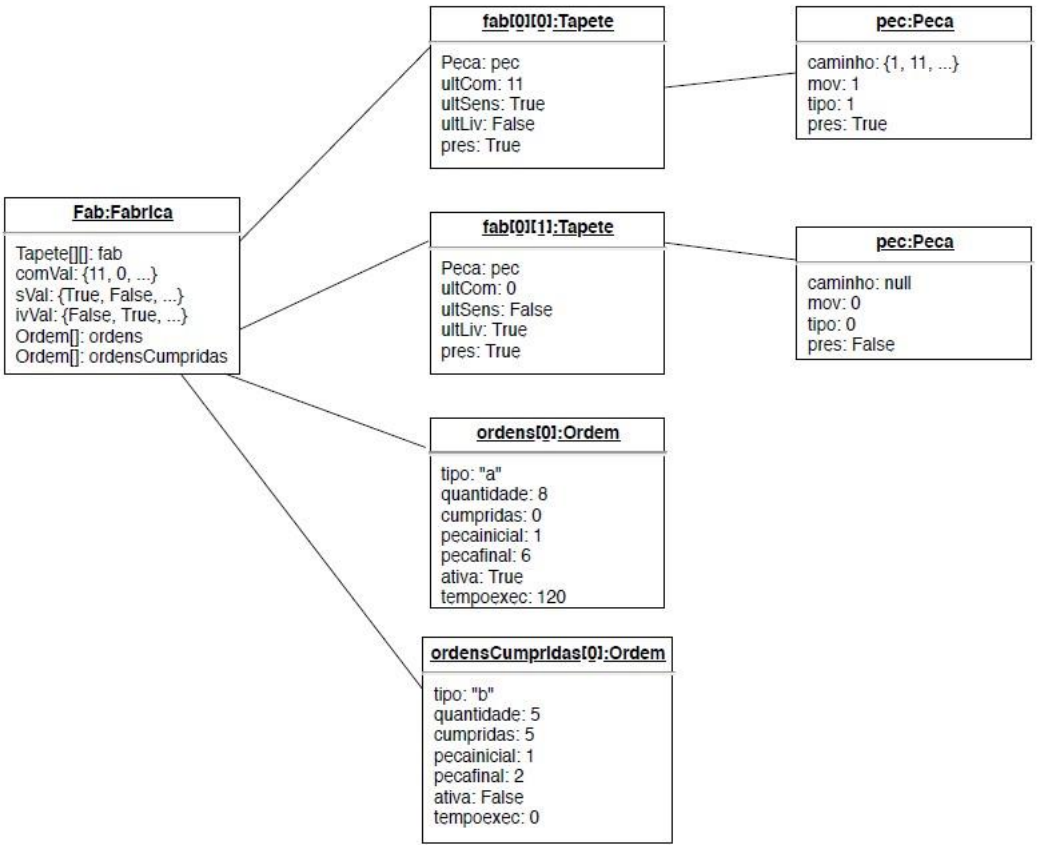
Diagrama de Classes – inicial



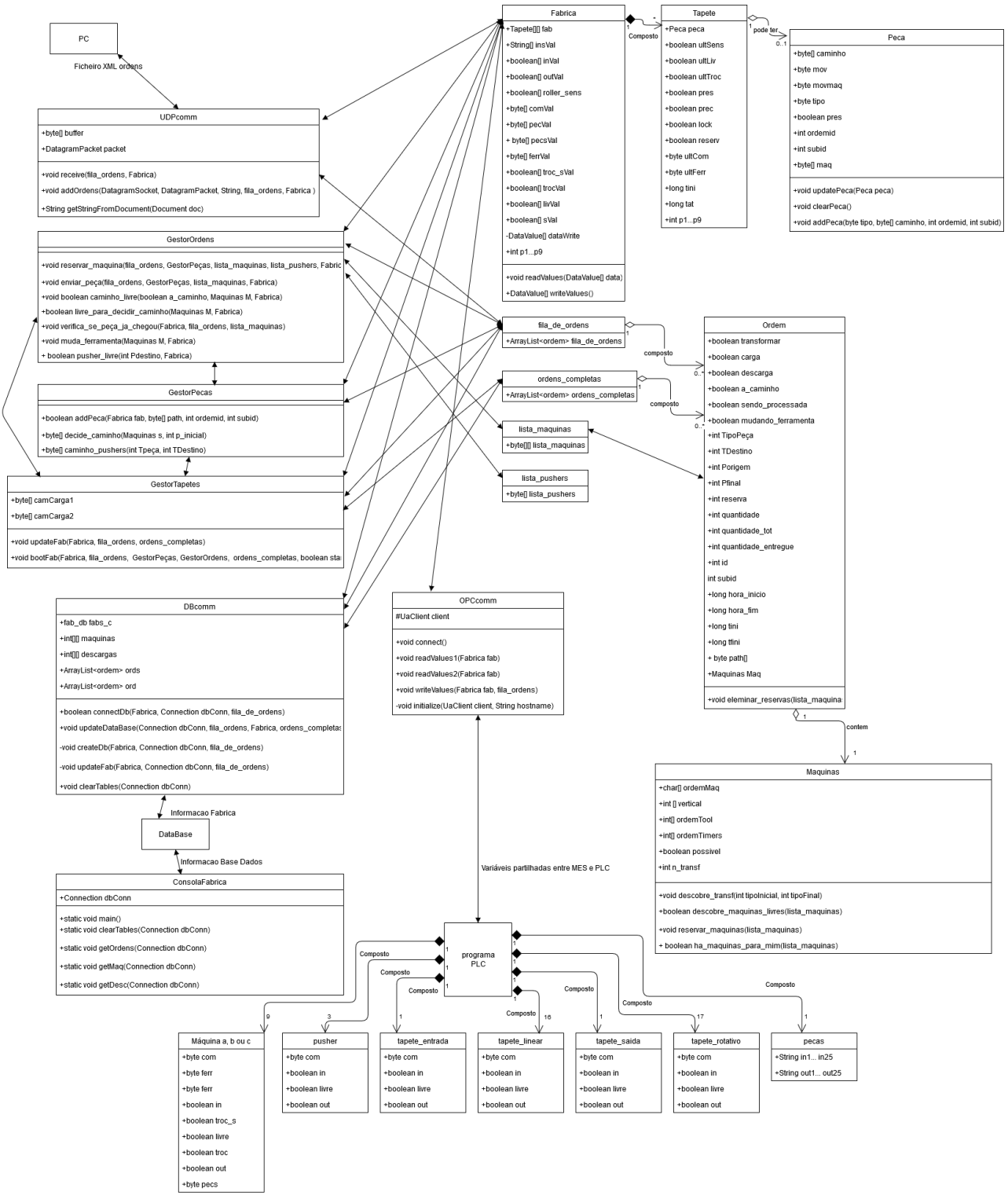
Sequence Diagram – inicial



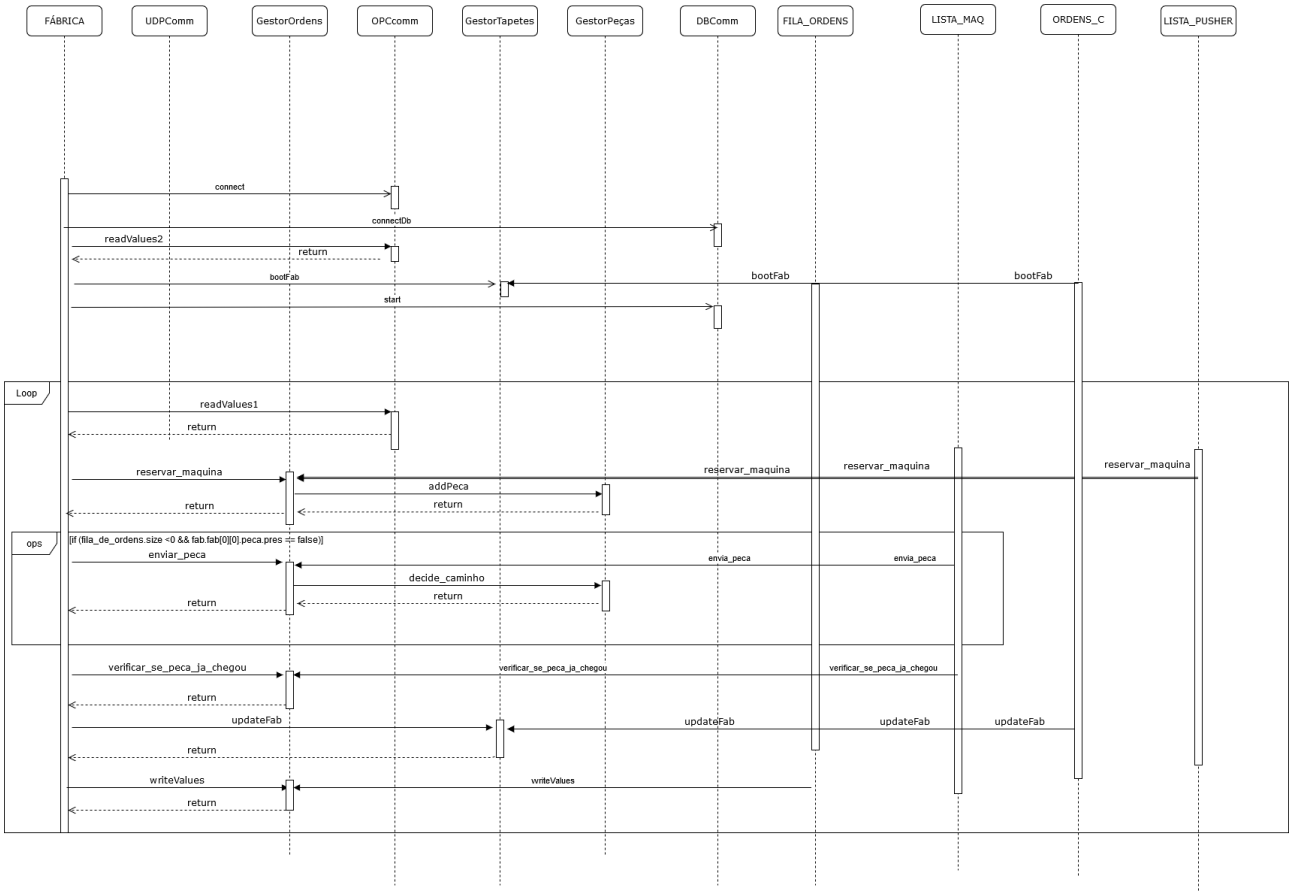
Object Diagram - inicial



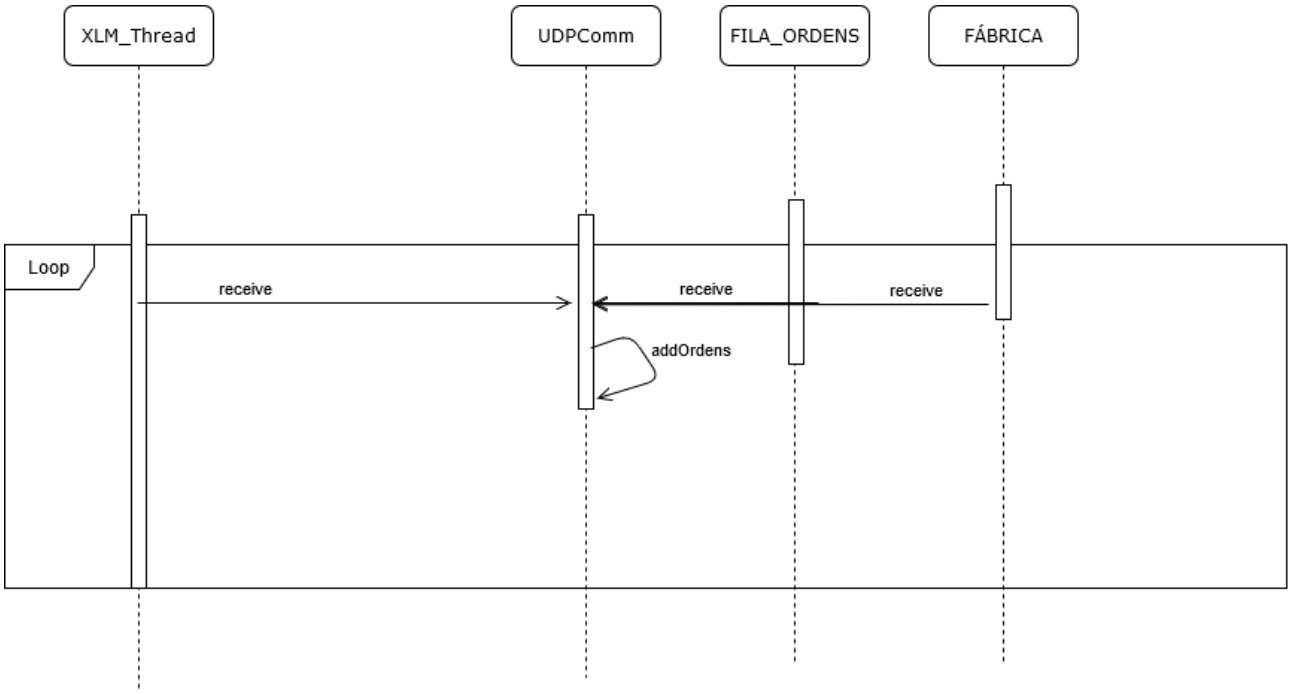
Class Diagram – final



Sequence Diagram – main Loop - final



Sequence Diagram – thread UDP - final



Sequence Diagram – thread DB - final