Desenvolvimento de servidor OPC UA para sistema CNC

Mestrado em Engenharia Mecânica – Produção Industrial

João Luís Fernandes Lucas

Leiria, setembro de 2019

Desenvolvimento de servidor OPC UA para sistema CNC

Mestrado em Engenharia Mecânica – Produção Industrial

João Luís Fernandes Lucas

Trabalho de Projeto realizado sob a orientação do Professor Doutor Carlos Neves

Leiria, setembro de 2019

Originalidade e Direitos de Autor

O presente relatório de projeto é original, elaborado unicamente para este fim, tendo sido devidamente citados todos os autores cujos estudos e publicações contribuíram para o elaborar.

Reproduções parciais deste documento serão autorizadas na condição de que seja mencionado o Autor e feita referência ao ciclo de estudos no âmbito do qual o mesmo foi realizado, a saber, Curso de Mestrado em Engenharia Mecânica – Produção Industrial, no ano letivo 2018/2019, da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria, Portugal, e, bem assim, à data das provas públicas que visaram a avaliação destes trabalhos.

Dedicatória

Para Ricardo,

Agradecimentos

Agradeço ao Professor Doutor Carlos Neves, pela disponibilidade, orientação e apoio que me deu ao longo deste percurso, sem o qual, certamente não teria sido possível concluir este trabalho.

Aos meus colegas, pelas suas contribuições para a realização deste trabalho, em especial a André Martins, pela ajuda imprescindível.

À minha família pelo apoio incondicional, e as bases que me deram para que pudesse sempre continuar. Obrigado Joaquim Lucas e Teresa Lucas.

Aos meus amigos, pelo incentivo e apoio em todos os momentos. Por todas as conversas, ou gestos reconfortantes, obrigado Sérgio Teixeira.

À minha amiga e companheira, por me equilibrar e suportar durante os momentos mais difíceis, e por celebrar os mais divertidos, obrigado Elsa Pedro.

Este trabalho foi desenvolvido no âmbito do projeto Mobilizador TOOLING4G, cofinanciado pelo Programa Operacional Competitividade e Internacionalização (POCI), através do Portugal 2020 e do Fundo Europeu de Desenvolvimento Regional (FEDER).

Resumo

A Indústria 4.0 é um conceito cada vez mais popular, e o que traz de novo para a indústria são novas formas de se olhar para conceitos como, por exemplo, a interoperabilidade de sistemas. Neste sentido, surgem tecnologias como a norma de comunicação OPC UA, que pretende ser uma resposta para a unificação das formas de comunicar entre sistemas de plataformas diferentes. Um dos tipos de sistemas utilizados na indústria, é do tipo CNC, onde já existem várias soluções ao nível do seu controlo.

Como objetivo deste trabalho, surge então, a adaptação de um servidor OPC UA a um sistema CNC já existente. Como base, decidiu-se utilizar uma solução da fundação OPC, por ser de acesso livre, código aberto e por ser uma ferramenta de referência, ligada diretamente a quem criou e gere esta tecnologia.

O processo começa por compreender de que forma foi construída a solução da norma neste módulo de software. Numa segunda fase, procurou gerar-se um sistema mais abrangente, que pudesse ser aplicado em vários equipamentos, de distintos fabricantes, tendo-se optado por construir um tipo abstrato de informação, uma classe genérica, que permita facilmente utilizar vários controladores, através da implementação em classes derivadas específicas. É, assim, elaborado um modelo de informação, seguindo as diretrizes da norma, para depois se integrar no servidor. Desenvolve-se a última funcionalidade, com a implementação de métodos para atualizar o modelo integrado.

O resultado é uma aplicação que permite disponibilizar a informação de um sistema CNC utilizando a norma OPC UA. O processo escolhido, neste trabalho, permite encontrar uma solução em que é possível aplicar a tecnologia de comunicação em sistemas que atualmente operam na indústria que, podendo ser heterogéneos, podem beneficiar da utilização de uma classe genérica, o que torna as implementações mais práticas.

Palavras-chave: Indústria 4.0, OPC UA, Interoperabilidade, CNC

Abstract

Industry 4.0 is an increasingly popular concept bringing to industry new ways to look into concepts like, for instance, the interoperability between systems. In this scope, new communications technologies emerge, such as the OPC UA norm, posed to be an answer to the unification of the way industrial system built with different platforms communicate. One of the types of systems used in industry is the CNC machine, where several controller technologies and brands coexist in the market.

The objective of the current work is to develop an OPC UA server application to be used with existing CNC machines. As a base, the solution chosen comes from the OPC Foundation, both because it is a free access open source solution and because it is a reference implementation, developed by the same entity that created and manages the technology.

The processes starts by understanding how this base implementation of the norm was structured and built. In a second phase, a server was developed so that it could be applied to different controllers from different manufacturers. The option taken was to build an abstract data type, a generic class, to accommodate the different controllers by implementing specific derived classes. In consequence, an information model following the guidelines of the norm is built to integrate the server's address space.

The result is an application that makes available, through the OPC UA norm, the relevant information present in a CNC controller. The approach chosen in this work, results in a software solution allowing the implementation of OPC UA to be applied in presently operating industrial CNC controllers which, although being heterogeneous, can benefit from the use of a generic class, turning practical implementations easier.

Keywords: Industry4.0, OPC UA, Interoperability, CNC

Índice

Originalidade e Direitos de Autor ... **iii**

Dedicatória ... **iv**

Agradecimentos ... **v**

Resumo ... **vi**

Abstract ... **vii**

Lista de figuras ... **x**

Lista de siglas e acrónimos ... **xii**

1. Introdução ... **1**

2. Enquadramento ... **2**

2.1. Enquadramento Histórico ... **2**

2.2. 4ª Revolução Industrial ... **3**

2.2.1. Sistemas Ciber-Físicos ... 4

2.2.2. Internet das Coisas ... 6

2.2.3. Big Data ... 6

2.2.4. RAMI 4.0 ... 7

2.2.5. Consola de administração de ativos ... 10

3. Tecnologias ... **12**

3.1. OPC UA ... **12**

3.2. Modelos de Arquitetura ... **13**

3.2.1. Servidor ... 13

3.2.2. Cliente ... 14

3.2.3. Cliente-Servidor ... 14

3.2.4. Modelo de informação em OPC UA: o *Address Space* ... 15

3.3. Modelo de Informação ... **17**

3.3.1. *Companion Specification* ... 18

3.3.2. *Information Model for CNC Systems* ... 18

4. Desenvolvimento de servidor OPC UA ... **20**

4.1. Introdução ... **20**

4.2. Aplicação sistema CNC ... **21**

4.2.1. Eding CNC ... 21

4.2.2. Software Eding ... 21

4.2.3. Classe genérica 24

4.3. Aplicação OPC UA 25
4.3.1. Introdução 25
4.3.2. Servidor Base 26
4.3.3. *Address Space* 27

4.4. Modelação do *Address Space* para o sistema CNC 28

4.5. Integração no servidor OPC UA 31
4.5.1. Atualização do *Address Space* 31

4.6. Resultados 31

5. Conclusão e trabalhos futuros 33

Bibliografia ou Referências Bibliográficas 34

Anexos 35

Anexo A - Servidor OPC UA para sistema CNC 35

Lista de figuras

Figura 1 - Mapa conceptual de um sistema Ciber-Físico (Lee, et al.).....5

Figura 2 – A evolução da *'Internet of Things'* (Evans, 2011).....6

Figura 3 - *Reference architecture model Industrie4.0* (RAMI4.0) (Adolphs, et al., 2016).....8

Figura 4 - Exemplos de componentes I4.0 (Adolphs, et al., 2015).....10

Figura 5 – Estrutura de uma consola de administração de ativos (Adolphs, et al., 2016).....11

Figura 6 – Organização da estrutura da norma OPC UA (OPC Foundation, 2017).....13

Figura 7 – Arquitetura de um servidor OPC UA (OPC Foundation, 2017).....13

Figura 8 – Arquitetura de cliente OPC UA (OPC Foundation, 2017).....14

Figura 9 – Arquitetura de um sistema OPC UA (OPC Foundation, 2017).....14

Figura 10 - *UA Object Model* (OPC Foundation, 2017).....15

Figura 11 - Atributos obrigatórios e opcionais (OPC Foundation, 2017).....16

Figura 12 - Diagrama de tipos de classe.....17

Figura 13 - Infraestrutura OPC UA (OPC Foundation, 2006).....17

Figura 14 - Origem de um *Address Space*.....18

Figura 15 – Arquitetura de sistema integrado OPC UA (VDW & OPC Foundation, 2017).....19

Figura 16 – Arquitetura de sistema desacoplado OPC UA (VDW & OPC Foundation, 2017).....19

Figura 17 - Diagrama de objetivo do projeto.....21

Figura 18 - Interface gráfico do *software* Eding.....22

Figura 19 - Esquema de funcionamento da tecnologia Eding (Eding, 2018).....23

Figura 20 - Interação dos componentes.....23

Figura 21 - Esquema de construção da classe genérica.....25

Figura 22 - Árvore de projetos exemplo da *stack* OPC UA.....25

Figura 23 - *Reference Server*.....26

Figura 24 - *Reference Client.*.....27

Figura 25 - *Namespaces* a utilizar num servidor para sistemas CNC (VDW & OPC Foundation, 2017).....28

Figura 26 - Exemplo de modelo de informação para sistemas CNC (VDW & OPC Foundation, 2017).....29

Figura 27 - Aspeto gráfico do software UAModeler.....29

Figura 28 - Estrutura do *AdressSpace*. 30

Lista de siglas e acrónimos

API	*Application Programming Interface*
A&E	*Alarms & Events*
CNC	Comando Numérico Computorizado
COM	*Component Object Model*
DCOM	*Distributed Component Object Model*
ESTG	Escola Superior de Tecnologia e Gestão
HDA	*Historical Data Access*
IPL	Instituto Politécnico de Leiria
OLE	*Object Linking and Embedding*
OPC	*Open Platform Comunications*
RAMI 4.0	*Reference Architecture Model Industrie* 4.0
SOA	*Service-Oriented Architecture*
UA	*Unified Architecture*
URI	*Uniform Resource Identifier*
VDE	*Verband Deutscher Elektrotechniker*
VDI	*Verein Deutscher Ingenieure*
VDW	*Verband der Wohnungswirtschaft*
XML	*Extensible Markup Language*
ZVEI	*Zentralverband Elektrotechnik – und Elektronikindustrie*

1. Introdução

Quando o mundo atravessa um período de rápidas inovações tecnológicas, e quando cada vez é mais comum a utilização de termos como Internet das Coisas ou Indústria 4.0, torna-se interessante a abordagem destes tópicos de modo a que estes conceitos se vão tornando aos poucos uma realidade mais presente. A aplicação de novas tecnologias terá os seus impactos em vários aspetos da sociedade, com a vinda de sistemas inteligentes que permitirão que a interação homem-máquina tome outros contornos. Como as tecnologias atuais estão a ficar mais acessíveis ao longo do tempo, também a sua aplicação se está a tornar mais fácil, o que leva a uma necessidade de adaptação mais rápida para manter os níveis de competitividade.

Este trabalho aborda uma forma de levar um sistema, já existente, no caminho idealizado pela Indústria 4.0, com a utilização das tecnologias que estão a ser desenvolvidas atualmente. O objetivo consiste na aplicação de uma tecnologia de comunicação e modelação de informação que torne um sistema CNC mais próximo de um sistema que possa ser reconhecido e operado em conjunto com outros sistemas. Neste caso específico, construiu-se uma aplicação da tecnologia OPC UA num equipamento CNC com um controlador construído pela empresa Eding CNC.

Dividiu-se este documento em cinco capítulos. Esta pequena introdução consiste no primeiro, o segundo contém alguns factos históricos e uma descrição de conceitos relacionados com a Indústria 4.0 que complementam o enquadramento. O terceiro capítulo pretende apresentar as tecnologias que serviram para sustentar o quarto, onde é descrito o processo seguido na construção de um servidor OPC UA que disponibiliza a informação contida num controlador de uma máquina CNC. Nesta parte do documento, estão também descritos os resultados obtidos. Por fim, no último capítulo, apresentam-se as conclusões do trabalho e deixam-se algumas ideias para dar continuidade a esta abordagem.

2. Enquadramento

2.1. Enquadramento Histórico

A primeira revolução industrial começa em Inglaterra, entre os anos de 1760 e 1840, e é caracterizada pela construção de caminhos de ferro e a invenção do motor a vapor. Durante este intervalo de tempo, as tecnologias utilizadas, tais como moinhos hidráulicos, moinhos eólicos e a força dos animais, começam a ser substituídos por motores a vapor, que viriam a permitir um aumento na eficiência das indústrias, como, por exemplo, as minas de carvão.

No final do século XIX, início do século XX, a eletricidade e a linha de montagem vêm revolucionar a indústria permitindo a produção em massa. Nesta altura, surgem barcos com motores a vapor, que permitem um avanço no transporte de mercadorias. A descoberta de novos combustíveis como o petróleo e o aparecimento do primeiro motor de combustão também descrevem aquela que foi a segunda revolução industrial.

A terceira revolução industrial, também chamada revolução técnico-científica, é assim conhecida pelos grandes avanços nas áreas da informática, telecomunicações, biotecnologia, robótica. Começa por volta da década de 1960, com o desenvolvimento dos semicondutores, passando pelos primeiros computadores de grande porte até aos computadores pessoais. Fica também conhecida pelo aparecimento da Internet na década de 1990, de grande importância pois permitiu ao mundo uma maior proximidade.

No século XXI assistimos ao que se acredita ser a quarta revolução industrial. É caracterizada por uma internet mais móvel e globalizada, por sensores mais pequenos e potentes que se tornaram mais baratos, pela inteligência artificial e pela aprendizagem automática. De acordo com Klaus Schwab (Schwab, 2016), a crescente sofisticação e integração das tecnologias digitais está a levar a uma transformação da sociedade e da economia mundial, o que leva os professores Erin Brynjolfsson e Andrew Mcafee, do *Massachusetts Institute of Technology* (MIT) a referirem-se a este período como a segunda idade da máquina. Já na Alemanha, em 2011 na feira de Hannover, o termo “Industry 4.0” foi apresentado e debatido para descrever como a organização da cadeia global de valor iria ser revolucionada. O mesmo autor acredita que esta revolução industrial terá o mesmo impacto que as anteriores, porém, refere que dois fatores lhe podem limitar o potencial. Um, serão os níveis de liderança e compreensão das mudanças futuras, o outro, um discurso que

descreva os desafios e oportunidades desta revolução, de modo a evitar uma reação fundamentalista contra mudanças basilares a caminho (Schwab, 2016).

2.2. 4ª Revolução Industrial

O mundo está a assistir a grandes alterações. As inovações tecnológicas estão a permitir que este esteja mais próximo: é possível hoje, por exemplo, aceder com facilidade a uma loja que esteja no outro lado do planeta. Tecnologias como a Internet ou sensores não são novas, o que surge e que torna este conceito de uma nova revolução interessante é o facto destas tecnologias poderem ser utilizadas em sistemas integrados, permitindo assim uma maior recolha de dados que facilita a gestão dos recursos e, consequentemente, do negócio (Schwab, 2016).

Vários países estão a abordar esta temática de forma semelhante criando os seus programas de desenvolvimento que contêm a estratégia e missão que melhor se adequam às suas economias, como exemplos, na Alemanha é a "Industrie 4.0", nos Estados Unidos da América chamaram-lhe *"Industrial Internet of Things"* e fundaram a *"Industrial Internet Consortium"*, em França a *"Alliance Industrie du Futur"*, na China o *"Made in China 2025"* ou, no caso do Japão, a *"Society 5.0"*. Estes são alguns exemplos de programas que abordam esta temática e têm por base características comuns, como a interoperabilidade, que consiste na capacidade de sistemas comunicarem entre si de forma a alcançar o mesmo objetivo. A virtualização, em que cada elemento é representado digitalmente e atualizado com informação dada por sensores ou pela rede. A descentralização, que se refere à capacidade de o elemento raciocinar e decidir de forma autónoma e ainda a orientação para serviços, retirando o foco da propriedade e fornecendo um conjunto de serviços individualizados.

Estes objetivos são alcançados tendo por base várias tecnologias que facilitarão o cumprimento das características mencionadas. São disso exemplo os sistemas ciber-físicos que reúnem os vários domínios das tecnologias: o físico, mecânico e eletrónico, por exemplo, o digital ou virtual e uma rede de comunicações. A realidade aumentada, que permitirá uma maior acessibilidade à informação sobre objetos reais, como, por exemplo, o caso de um operador que pretenda realizar manutenção a uma máquina e que, através de um dispositivo, possa aceder aos componentes presentes na máquina e tomar conhecimento dos métodos necessários para realizar a sua tarefa. A produção aditiva, que permite a prototipagem rápida de forma a testar um produto para o mercado e que, em alguns casos, é

já utilizada para produzir componentes para utilização industrial. A computação em nuvem também é uma tecnologia que se prevê de grande utilização nesta nova fase, permitindo que a informação seja armazenada e tratada a partir de uma base comum. A cibersegurança tem um papel importante em prevenir manipulação em informação sensível: por exemplo, um sistema de travagem de um veículo não pode ser acedido por elementos externos por haver risco de vida. A análise de *Big Data*, que permite o tratamento da enorme quantidade de dados disponível e que leva a que sistemas e recursos tenham de ser definidos para esse fim. A aprendizagem automática e a inteligência artificial que permitirão a robótica autónoma e colaborativa de forma a otimizar os sistemas (Gilchrist, 2016).

A utilização destas tecnologias vai além dos avanços que proporcionará à indústria, já que também a sociedade beneficiará destes novos conceitos e o impacto poderá ser sentido em várias áreas. Klaus Schwab descreve o impacto na economia como "monumental" e foca dois aspetos principais: o crescimento e a empregabilidade. Para o crescimento, defende que os países e as empresas devem estar sempre atualizados nas inovações, e que as estratégias, devem focar a oferta de produtos e serviços em vez de adotar a redução de custos, por serem de menor eficácia. O caso da empregabilidade, já vem sendo discutido desde que os primeiros robots começaram a integrar as linhas de produção e aos poucos a substituir humanos em tarefas repetitivas. O impacto agora prevê-se ser maior pela velocidade a que as inovações tecnológicas estão a ocorrer e será necessária uma atenção cuidada a efeitos menos favoráveis neste sentido (Schwab, 2016).

Estas tecnologias permitirão fábricas inteligentes que se podem adaptar às exigências do mercado. Pela sensorização dos produtos, o cliente terá um papel mais preponderante na forma como estes evoluem, a produção terá tendência para ser mais personalizada e os produtos a terem um ciclo de vida mais curto, o que obrigará ao ajuste dos modelos de negócio. A manutenção preventiva, apoiada no conhecimento gerado, também promete uma melhoria na eficiência, gerindo as intervenções na linha de produção (Schwab, 2016).

2.2.1. Sistemas Ciber-Físicos

Os sistemas ciber-físicos são uma das bases desta revolução industrial, Edward Lee e Sanjit Seshia descrevem estes sistemas como a integração da computação, de uma rede de comunicações e de processos físicos, geralmente com retorno de informação, em que os processos físicos afetam os computacionais e o inverso também se verifica (Lee, et al., 2011).

Pela definição de sistema ciber-físico, tudo o que contiver computação integrada, uma rede de comunicação e processos físicos, pode ser considerado como tal. Alasdair Gilchrist chega a dar exemplo extremo de um operador humano como um sistema ciber-físico, sendo que o cérebro seria a sua unidade computacional, a comunicação é realizada através dos sistemas do corpo humano utilizados para comunicar e a interação física é feita utilizando, por exemplo, as mãos. Então, da mesma forma, uma máquina que contenha uma unidade de computação, que esteja ligada à rede e que atue consoante a informação que é partilhada, será considerada um sistema com esta definição (Gilchrist, 2016).

Como estes sistemas terão uma elevada complexidade, há vários requisitos a ter em conta na hora da sua conceção. A segurança é um dos pontos fundamentais, pois é necessário salvaguardar diferentes aspetos neste âmbito: há a segurança de operação *(safety)* do sistema, como por exemplo, o sistema tem de permitir intervenção quando for pedido, prever acidentes para quem o opera e para os outros sistemas circundantes. Há também a segurança*(security)* do próprio sistema, em que é necessário protegê-lo de forma a poder estar ligado à rede sem sofrer manipulação externa (Gilchrist, 2016). Na Figura 1, podemos ver um diagrama que pretende descrever uma estrutura de um sistema ciber-físico, mostrando o seu conteúdo, o que o caracteriza e define.

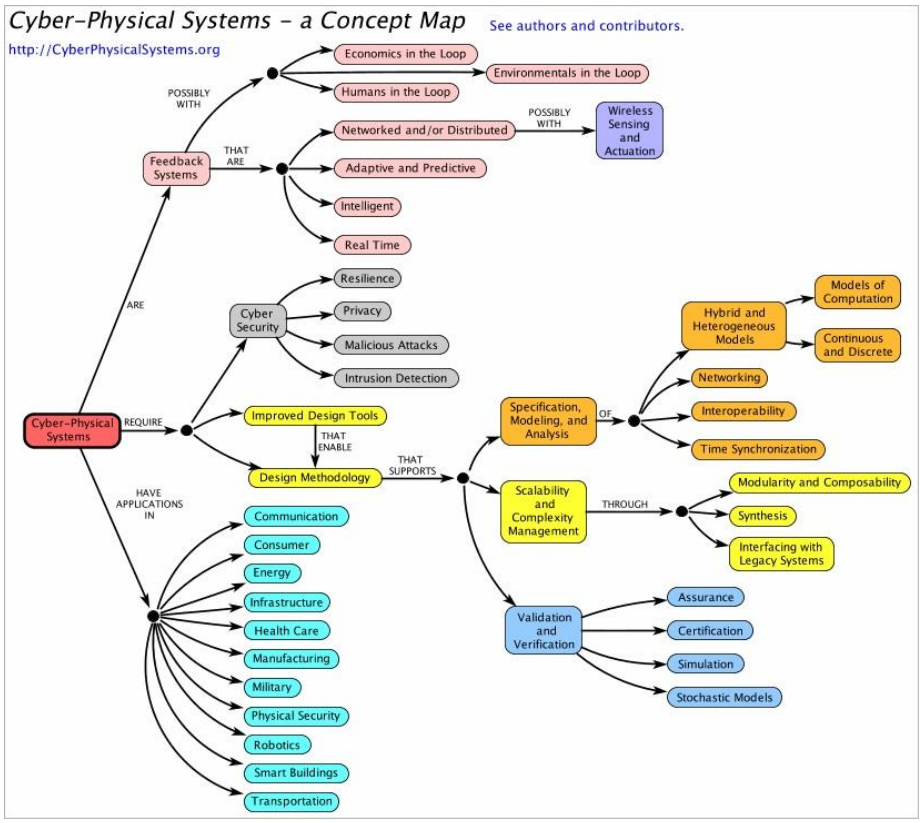


Figura 1 - Mapa conceptual de um sistema Ciber-Físico (Lee, et al.).

2.2.2. Internet das Coisas

A Internet das Coisas é um termo que tem sido popularizado nos últimos tempos, sendo um conceito que se refere à comunicação entre quase tudo o que possa ser monitorizado, tanto objetos físicos como virtuais. Esta tecnologia é promissora no aspeto em que permite que os dados gerados sejam convertidos em informação útil, isto quer dizer que a facilidade na obtenção de conhecimento sobre determinados sistemas é maior.

Na Figura 2 podemos ver um gráfico que representa o número de objetos ligados à rede face à população mundial ao longo do tempo. Nele, a Cisco Internet Business Solutions Group (IBSG), em 2011, previa que, em 2020, o número total de objetos ligado à Internet seria de 50 mil milhões (Evans, 2011).

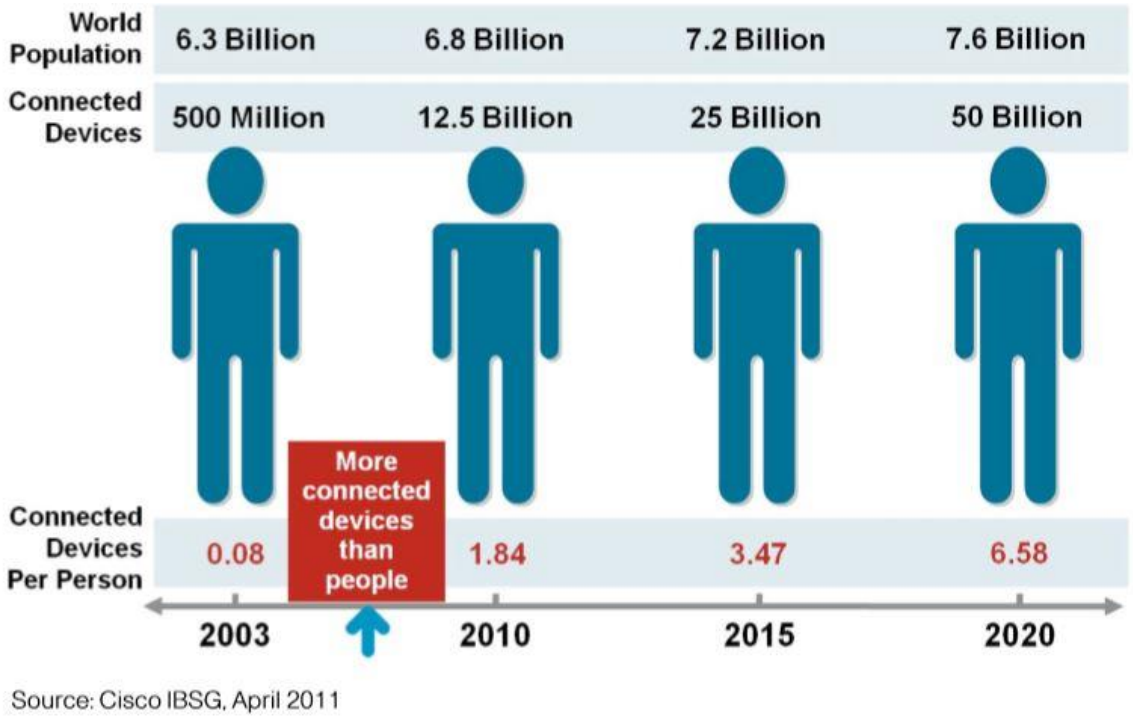


Figura 2 – A evolução da *'Internet of Things'* (Evans, 2011).

2.2.3. Big Data

O conceito de *Big Data* surge para se referir às enormes quantidades de dados gerados pelos sistemas que estão ligados à rede. Segundo Alasdair Gilchrist, estes dados podem ser estruturados ou não-estruturados, o que aumenta a dificuldade de análise por parte das tradicionais bases de dados. Para solucionar este problema as indústrias já podem contar com serviços de armazenamento em nuvem para gerir *Big Data* com capacidades elevadas de armazenamento (Gilchrist, 2016).

O mesmo autor também divide os tipos de *Big Data* de acordo com várias características. Começando pelo volume, onde uma quantidade de dados é mais fiável quantos mais forem os seus constituintes. A velocidade, que está ligada com a capacidade de resposta do sistema, ou seja, a velocidade a que os dados são recebidos e a rapidez com que são analisados. A variedade, que se refere ao facto de os dados serem provenientes de várias fontes, leva à necessidade da sua organização prévia à análise. Outra característica importante é a veracidade dos dados, por haver necessidade de, após a recolha e armazenamento, efetuar uma validação. Refere ainda o valor que os dados representam para o negócio em questão e, por último, a visibilidade, que permite a compreensão de modas e correlações entre dados (Gilchrist, 2016).

2.2.4. RAMI 4.0

Na procura por um modelo que pudesse implementar as tecnologias da Indústria 4.0, várias iniciativas foram tomadas em forma de modelos de arquitetura de referência. Estes modelos descrevem uma estratégia base para a arquitetura a usar em sistemas de informação em contexto de produção industrial. Talvez dois dos modelos mais difundidos, sejam o *Industrial Internet Reference Architecture* (IIRA) pelo *Industrial Internet Consortium* (IIC), e o RAMI4.0 pela Plattform Industrie 4.0, sendo que, nos países europeus, tem sido o modelo alemão o mais adotado, possivelmente por uma questão de proximidade com a fonte.

Para responder às necessidades da Indústria 4.0, as associações VDI/VDE-IT e ZVEI, na Alemanha, publicam em julho de 2015 um artigo chamado *"Reference Architecture Model Industrie 4.0 (RAMI 4.0)"* (Adolphs, et al., 2015), em que descrevem um modelo que permite integrar e assegurar que todos os participantes se compreendem entre si.

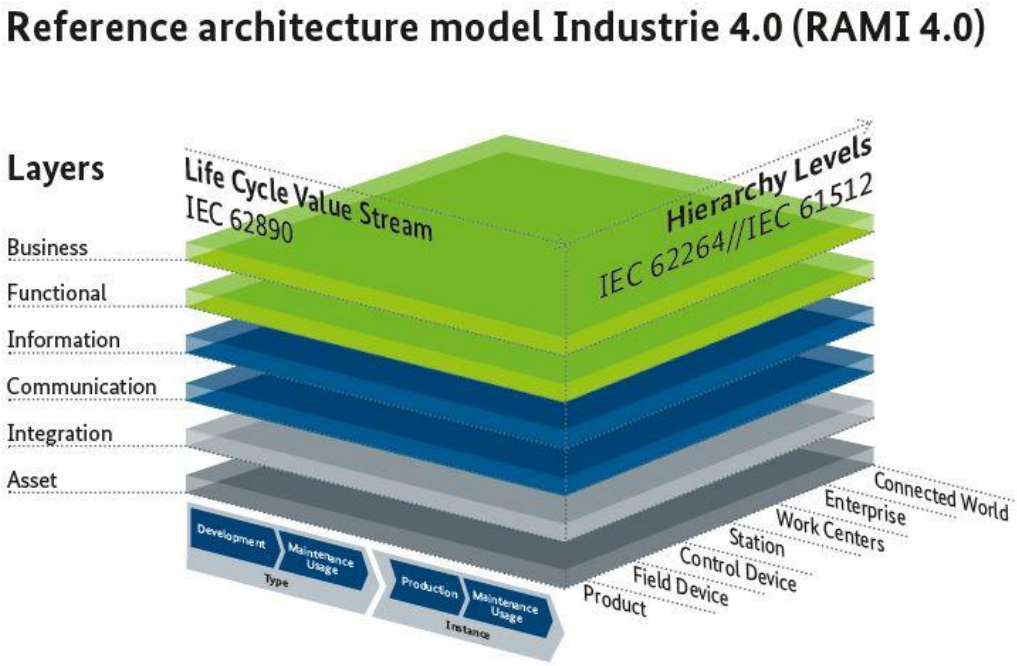


Figura 3 - *Reference architecture model Industrie4.0* (RAMI4.0) (Adolphs, et al., 2016).

A melhor forma que encontraram para representar esta arquitetura foi com um modelo a três dimensões, ilustrado na Figura 3, onde se podem ver camadas ao longo do eixo vertical que representam as várias áreas que precisam de resposta por parte das tecnologias de informação. No eixo horizontal da esquerda, estão representados, o ciclo de vida e a cadeia de valor do produto que pretendem descrever a forma como um produto é visto, consoante o seu estado de desenvolvimento. No eixo horizontal da direita estão representados os participantes, desde o produto até à rede global, onde são descritas as funcionalidades e responsabilidades numa hierarquia funcional (Adolphs, et al., 2015).

Os objetivos deste modelo, consistem em ser um modelo de arquitetura simples e organizável, conseguir a minimização do número de normas envolvidas, a identificação de relações e a definição de regras de alto nível.

No eixo vertical, onde estão representadas as camadas, começando pela do negócio, que pretende assegurar a integridade das funções na cadeia de valor, ter uma visão sobre o modelo de negócio e o resultado geral de todo o processo, criar regras que o sistema terá de seguir, organizar os serviços da camada funcional e fazer a ponte entre diferentes processos de negócio (Adolphs, et al., 2015).

A camada funcional tem objetivos como o de fazer uma descrição formal das funções e ser uma plataforma para uma integração horizontal das várias funções. É esperado que as regras

e as tomadas de decisão ocorram dentro desta camada de forma a que a integridade da informação no processo e a integração a nível técnico sejam asseguradas.

A camada de informação tem como objetivos criar um ambiente ativo para um pré-processamento de eventos e a execução de regras relacionadas com eventos, ou seja, servir como ponte entre a camada funcional e os dados recebidos da camada de comunicação.

A camada de comunicação faz a normalização da comunicação para que esteja num formato de dados inteligível para a camada da informação e também mantém uma coleção de serviços para controlo da camada de integração.

A camada de integração será responsável por providenciar a informação de objetos, tais como componentes físicos, documentos ou mesmo software para que possam ser processados por computador e pode também gerar eventos a partir dos objetos. Esta camada dará suporte computacional ao processo técnico.

A camada de ativos, representa a realidade, onde todos os objetos são considerados. Os humanos também farão parte desta camada onde comunicam com o mundo virtual através da camada de integração.

No eixo horizontal do ciclo de vida e cadeia de valor há uma distinção entre tipo e instância. O conceito de tipo será criado com a ideia inicial, ou seja, algo que está numa fase de desenvolvimento e que, só após uma fase de validação o tipo é disponibilizado para produção em série. No caso do conceito de instância, já se entende que é algo gerado com base no tipo genérico, em que, por exemplo, terá um único número de série. Os dados recolhidos a partir das instâncias podem originar correções nos tipos iniciais, dando lugar a novos tipos que, por sua vez, darão origem a novas instâncias com as alterações implementadas. Esta relação entre tipo e instância define a forma de como o produto evolui consoante os dados recolhidos ao longo da sua vida útil.

No eixo dos níveis de hierarquia, o modelo pretende descrever a interação entre participantes e a forma como eles comunicam entre níveis. As principais diferenças entre este modelo e a tradicional pirâmide da automação são dois novos níveis, um na base, o próprio produto, sendo ele mesmo capaz receber e de gerar informação, e outro no topo, com a introdução do nível que representa todo o resto do mundo ligado à rede.

Para a implementação deste modelo, é necessário transformar os objetos em componentes I4.0, e, para que sejam assim considerados, esses objetos têm de ser vistos como entidades e possuir capacidade de comunicação (ativa ou passiva), o que significa ter uma *Asset Adminstration Shell*, termo que, neste documento, se traduzirá como "consola de administração de ativos". A Figura 4 ilustra alguns exemplos de componentes I4.0.

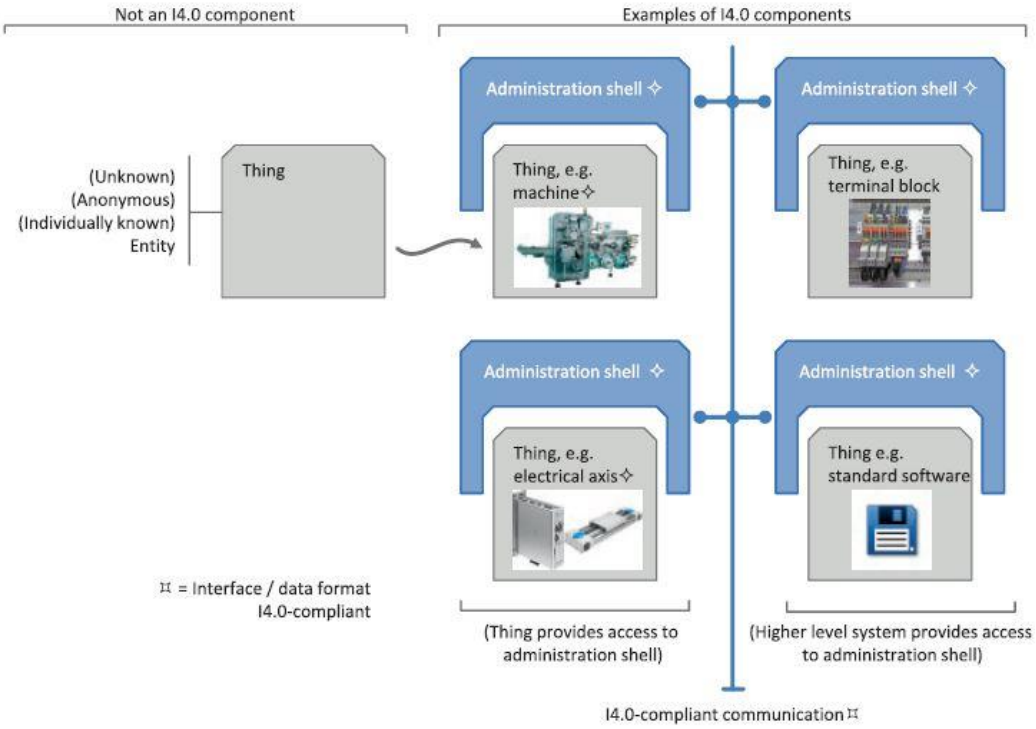


Figura 4 - Exemplos de componentes I4.0 (Adolphs, et al., 2015).

2.2.5. Consola de administração de ativos

Para enfrentar o desafio da Indústria 4.0, a interoperabilidade dos componentes I4.0, tendo em conta que se pode estar a falar de objetos tão distintos como uma máquina ou um software, é apresentado um modelo, que pretende assegurar essa comunicação entre componentes I4.0, partindo deste conceito de consola de administração de ativos (Adolphs, et al., 2016).

Como características gerais, um componente I4.0 tem de ser identificável, de forma unívoca, através de um identificador único (ID). A sua comunicação tem de ser baseada numa arquitetura orientada para serviços (SOA), como, por exemplo, OPC-UA. Tem de permitir ser operado por diferentes tipos utilizadores, eventualmente com permissões distintas. Por exemplo, no caso de um operador local o que lhe é permitido fazer será diferente do que estará acessível a um operador remoto. Tem também de ter a capacidade para fornecer a sua

descrição virtual, o que significa disponibilizar informação sobre as respetivas características, podendo ser, por exemplo, descrições das funções da máquina, os seus fluxos de trabalho e informações sobre os seus constituintes. Há ainda que garantir a proteção e segurança: cada componente deve manter o mínimo indispensável de segurança para manter as suas funcionalidades sem afetar outros. Tem de garantir também qualidade de serviços, onde devem ser tomadas em conta a interoperabilidade, capacidade de diagnóstico, o risco de falha ou a sincronização de relógios. O estado de cada componente I4.0 tem de estar sempre disponível, de forma a ser gerido local e globalmente para coordenação do fluxo de trabalho. Um componente I4.0 tem ainda de permitir o seu agrupamento com outros de forma a obter um único componente I4.0, esta característica é útil, por exemplo, no caso de uma máquina modular (Adolphs, et al., 2016).

Uma consola de administração de ativos é composta genericamente por um cabeçalho e um corpo. O cabeçalho deve conter uma lista de propriedades que permitam a identificação e descrição dos objetos geridos tal como os seus serviços e funções. O corpo será composto pelo gestor do componente que controlará eventuais submodelos, em que cada submodelo terá as suas propriedades individuais (Adolphs, et al., 2016). A Figura 5 ilustra a descrição feita.

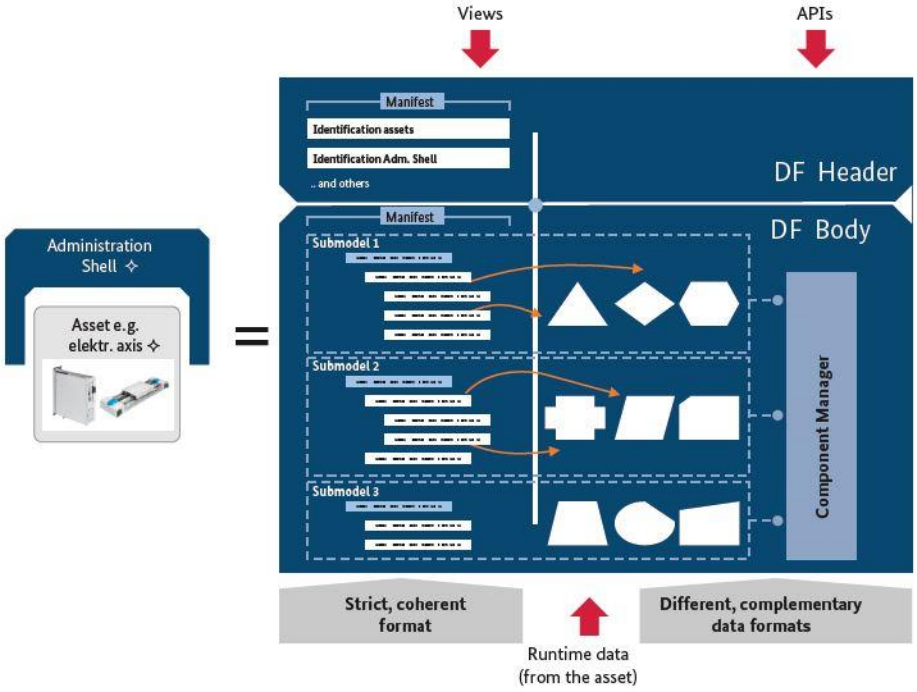


Figura 5 – Estrutura de uma consola de administração de ativos (Adolphs, et al., 2016).

3. Tecnologias

3.1. OPC UA

Nos anos 90, através da colaboração de várias empresas numa tentativa de encontrar uma forma de transmitir informação a partir de autómatos programáveis, surge a primeira versão simplificada da especificação OPC para *Data Access* (DA) que, de acordo com a fundação OPC (OPC Foundation, 2006), consiste num protocolo de comunicação que permite encapsular outros protocolos através de um sistema normalizado. Este utiliza uma biblioteca de comunicação de que a Microsoft é proprietária chamada COM/DCOM limitando-o a sistemas baseados em Windows. OPC no início significava OLE (*Object Linking and Embedding*) *for Process Control* e era apenas utilizado para acesso a dados (DA). Com as extensões HDA (*Historical Data Access*) e A&E (*Alarms and Events*) tornou-se possível guardar dados e identificar eventos discretos. Apesar de serem bastante utilizadas, estas especificações têm limitações, tais como restringir o funcionamento a sistemas baseado em Windows, a sua segurança tem defeitos e as especificações não partilham a mesma semântica, obrigando assim a ter vários servidores para cada equipamento se se pretender utilizá-las em simultâneo.

A OPC Foundation apresentou em 2006 a primeira versão da norma OPC UA (*Open Platform Comunication Unified Architecture*) que pretende ser o protocolo de comunicação normalizado para que vários tipos de sistemas possam trocar informação independentemente da sua plataforma. Este conceito é baseado numa arquitetura orientada para serviços (SOA) que integra todas as funcionalidades do OPC *Classic* e pretende ainda permitir que um único servidor forneça toda a informação e acesso aos serviços com segurança ativa. As características principais adicionadas neste modelo quando comparado com o anterior são a de não depender de um sistema operativo específico, ter segurança integrada através de um sistema *username-password* ou por certificados (X.509), ter uma função para descobrir outros servidores e clientes numa rede e de permitir a definição de semânticas normalizadas em áreas especificas.

A norma OPC UA é organizada em várias partes como mostra a Figura 6. As partes de 1 a 7 e 14 definem as capacidades centrais do OPC UA, as partes de 8 a 11 fazem a descrição de como estas capacidades se aplicam a vários tipos de acesso, a parte 12 é relativa aos

mecanismos de descoberta para OPC UA e a parte 13 mostra de que formas se podem agregar dados (OPC Foundation, 2017).

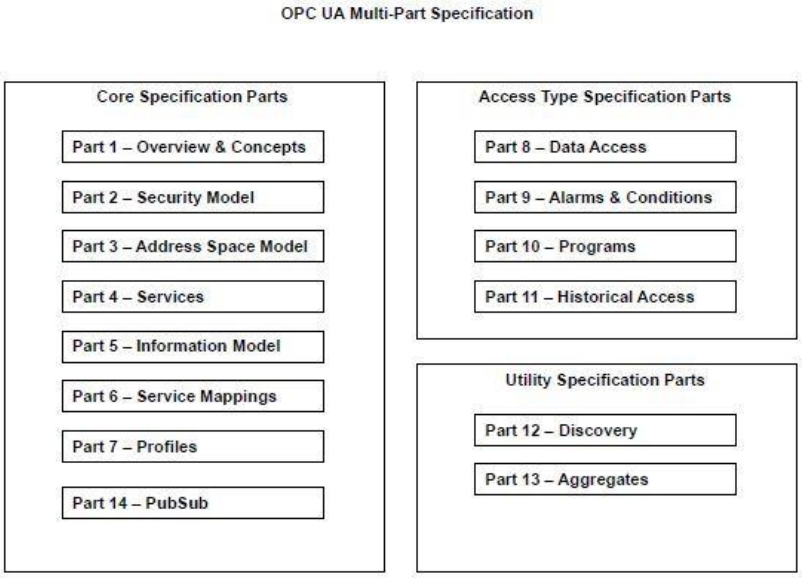


Figura 6 – Organização da estrutura da norma OPC UA (OPC Foundation, 2017).

3.2. Modelos de Arquitetura

3.2.1. Servidor

A Figura 7 mostra os elementos genéricos de um servidor OPC UA e de que forma eles se relacionam.

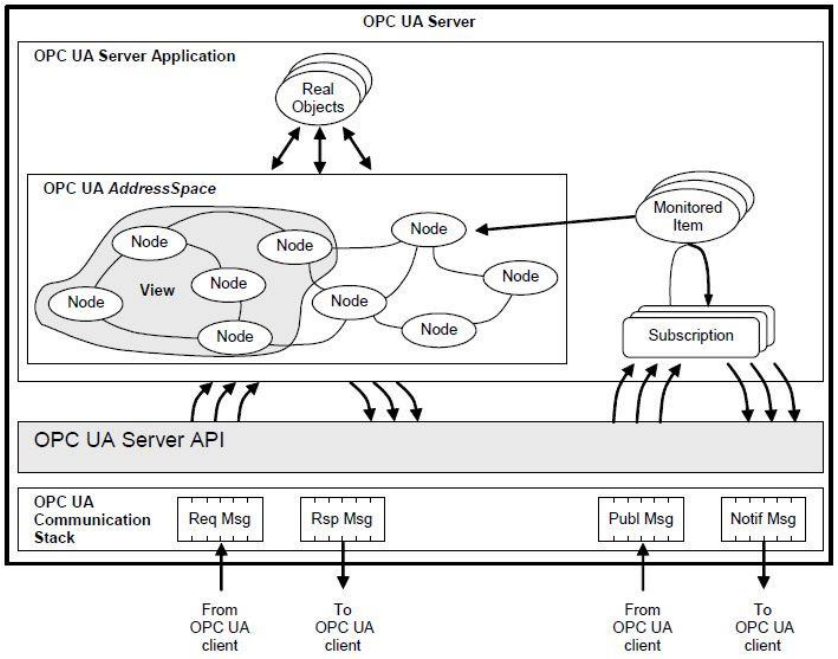


Figura 7 – Arquitetura de um servidor OPC UA (OPC Foundation, 2017).

Na Figura 7, o que está ilustrado como *"Real Objects"* refere-se aos objetos físicos ou software monitorizados/controlados pelo servidor. O servidor utiliza uma API para fazer a comunicação entre cliente e servidor através da *comunication stack.*

3.2.2. Cliente

O modelo de arquitetura de um cliente OPC UA onde são ilustrados os elementos principais e a relação entre eles está representado na Figura 8.

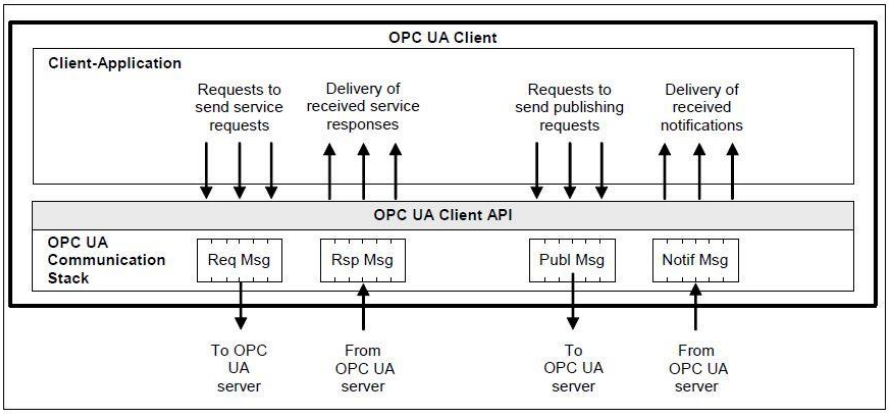


Figura 8 – Arquitetura de cliente OPC UA (OPC Foundation, 2017).

O item "*Client Application*" representa o código que implementa a função do cliente, o item "*Client API*" representa uma interface interna que é usada para isolar o código da communication stack. À semelhança do que acontece no caso do cliente, para fazer a comunicação com o servidor, a aplicação de cliente utiliza a API para fazer a interface com as funções implementadas na *communication stack* para enviar e receber informação para o servidor na forma de serviços OPC UA (descritos na parte 4 da norma).

3.2.3. Cliente-Servidor

A Figura 9 ilustra como uma aplicação OPC UA pode combinar os componentes Cliente e Servidor de forma a permitir que um ou mais clientes se possam ligar a um ou mais servidores.

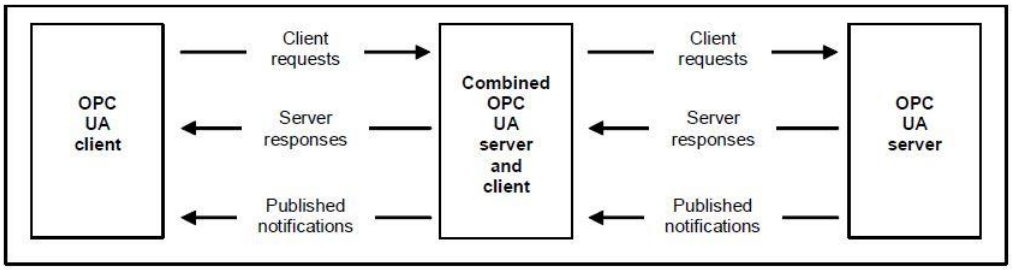


Figura 9 – Arquitetura de um sistema OPC UA (OPC Foundation, 2017).

3.2.4. Modelo de informação em OPC UA: o *Address Space*

O conceito de *"Address Space"*, fundamental neste paradigma, é definido como um conjunto de objetos a que os clientes podem aceder utilizando serviços de acordo com a norma, em particular da sua parte 3. O objetivo deste conceito é o de normalizar a forma como os servidores apresentam a informação para os clientes. Para isso, é apresentado um modelo de objeto que os define em termos de variáveis e métodos, conforme se vê na Figura 10, que mostra o modelo de objeto OPC UA. No *Address Space* estes elementos são representados como nós. Cada nó está associado a uma classe de tipo de nó que, por sua vez, serve para representar um tipo diferenciado de objeto (OPC Foundation, 2017).

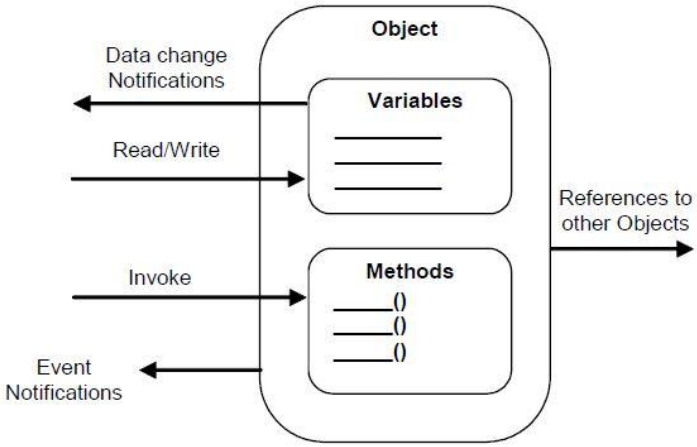


Figura 10 - *UA Object Model* (OPC Foundation, 2017).

As classes de nós são definidas por atributos e referências, devendo ser instanciadas quando um nó é definido no *Address Space*. Estas definições podem ser encontradas na parte 5 norma.

Os atributos consistem nos elementos de informação que descrevem os nós. São a estes que os clientes podem aceder utilizando os serviços OPC UA. Na definição das classes, os atributos podem ser de implementação obrigatória ou opcional o que leva a que nós da mesma classe possam ter um conjunto de atributos diferente consoante a sua aplicação/utilização, podemos ver uma ilustração do que foi descrito na Figura 11. Já no caso das referências são utilizadas para criar relações entre nós. Estas tomam vários tipos e podem, por exemplo, ser utilizadas para definir a cadeia hierárquica entre os nós.

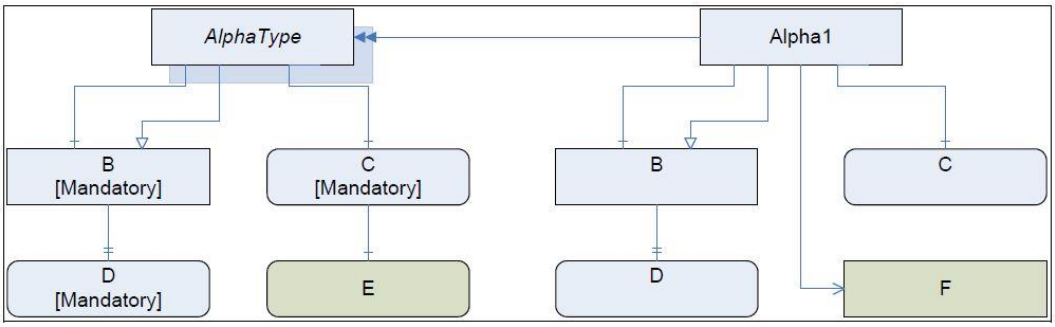


Figura 11 - Atributos obrigatórios e opcionais (OPC Foundation, 2017).

A norma define uma classe de nó base da qual todas as outras são derivadas: a "*Base NodeClass*". Existem três categorias de classes: para definição de instâncias, para definir tipos para essas instâncias e para definir os tipos de dados.

A classe base é composta por vários atributos, que podem ser de caráter obrigatório ou opcional. Esta classe tem quatro atributos obrigatórios na definição de um nó, que são:

NodeId – Identificador único do nó;

NodeClass – Identifica a classe do nó, por exemplo se se trata de um objeto, variável, método, entre outros;

BrowseName – Este atributo é utilizado como ajuda para que um operador humano possa ler quando percorre a árvore de um conjunto de nós;

DisplayName – Este é o atributo que os clientes devem utilizar quando é necessário mostrar o nome do nó ao utilizador.

No *Address Space* para que seja dada uma função a cada 'nó' é apresentado o seguinte conjunto de tipos de classe (*NodeClass*) (Postół, 2016):

View – Define um subconjunto de 'nós' dentro do *Address Space*, por exemplo para mostrar o conjunto de 'nós' mais relevante para o cliente;

Object – Representa sistemas, componentes, objetos físicos e objetos de software;

ObjectType – Fornece definições para a classe *Object*;

Variable – Utilizadas para conter informação sobre os objetos, como por exemplo um valor;

VariableType – É utilizado para fornecer definições de tipos para a classe *Variable*;

DataType – Define tipos de informação, simples e complexa, para os valores da classe *Variable*;

Method – Utilizada para definir funções executáveis;

ReferenceType – Serve para definir o significado das relações entre 'nós'.

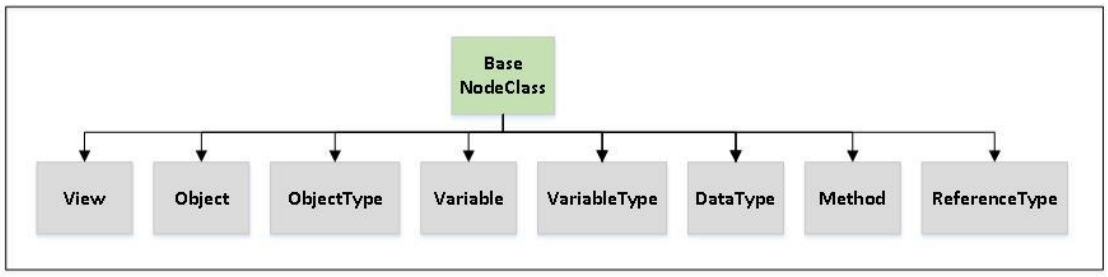


Figura 12 - Diagrama de tipos de classe.

A Figura 12 ilustra a relação dos tipos de classe com a origem. As definições destes tipos devem ser utilizadas quando o tipo de informação se espera que seja utilizado mais que uma vez no mesmo sistema ou para interoperabilidade entre sistemas diferentes que suportem a mesma definição de tipos (OPC Foundation, 2017).

3.3. Modelo de Informação

O modelo de informação é utilizado para descrever a normalização de 'nós' de um Address Space de um servidor (OPC Foundation, 2017). Segundo Mariusz Postół para tornar sistemas interoperáveis, a troca de informação deverá estar associada a um modelo representativo de informação, referindo que, em OPC UA, é utilizado um objeto como noção fundamental para representar a informação e comportamento de um sistema subjacente (Postół, 2016).

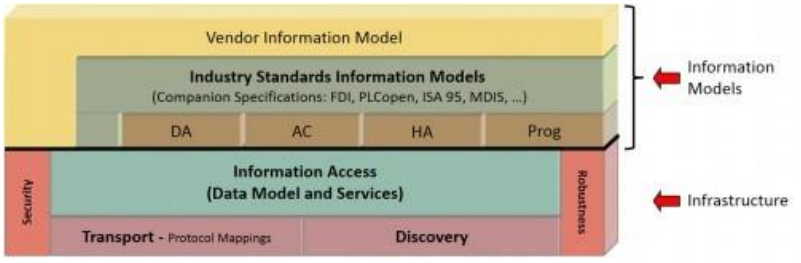


Figura 13 - Infraestrutura OPC UA (OPC Foundation, 2006).

Na Figura 13 está representada a infraestrutura OPC UA e de que forma se inserem os modelos de informação, sendo que o bloco denominado por "*Information Access*" se refere ao modelo de informação da norma.

3.3.1. *Companion Specification*

Os modelos de informação que se referem a problemas ou setores específicos da indústria são chamados "*Companion Specification*" e são criados em processos colaborativos entre os atores interessados num dado setor de atividade. Na Figura 14, está representada a origem de referências para a obtenção dos elementos de um *Address Space*.

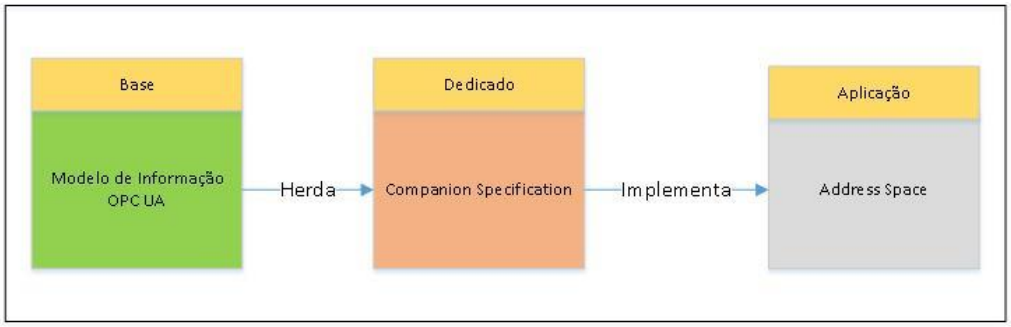


Figura 14 - Origem de um *Address Space*.

A OPC Foundation prevê três formas distintas de se produzirem estas especificações: internas, que são criadas através de grupos de trabalho internos na fundação, em parceria, onde há a possibilidade de criar um grupo de trabalho entre a OPC Foundation e outra organização e externas onde a fundação disponibiliza um template na web para ser descarregado.

3.3.2. *Information Model for CNC Systems*

Neste caso, é abordada uma *Companion Specification* em específico, a "*OPC UA Information Model for CNC Systems*", elaborada pela VDW e pela OPC Foundation. Esta consiste no modelo de informação específico para sistemas CNC. Aqui são definidos os *NodeIds*, *Browsenames*, os tipos de nós a utilizar para estruturar o modelo e ainda o tipo de dados a utilizar em cada nó.

A especificação prevê dois tipos de arquitetura de sistemas, do tipo integrado como ilustra a Figura 15 ou do tipo desacoplado como está na Figura 16. O tipo integrado refere-se a sistemas construídos de base com o protocolo OPC UA a integrar a programação. O tipo desacoplado refere-se a sistemas já existentes onde um componente servidor OPC UA faz a ligação com esse sistema e o liga à rede de trabalho da produção.

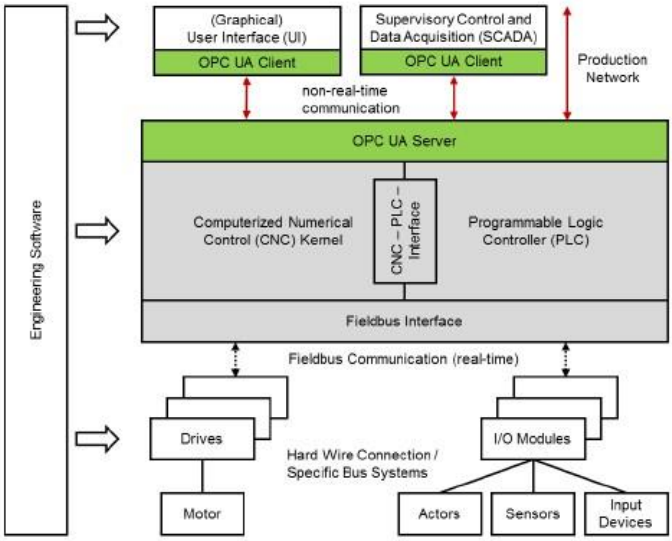


Figura 15 – Arquitetura de sistema integrado OPC UA (VDW & OPC Foundation, 2017).

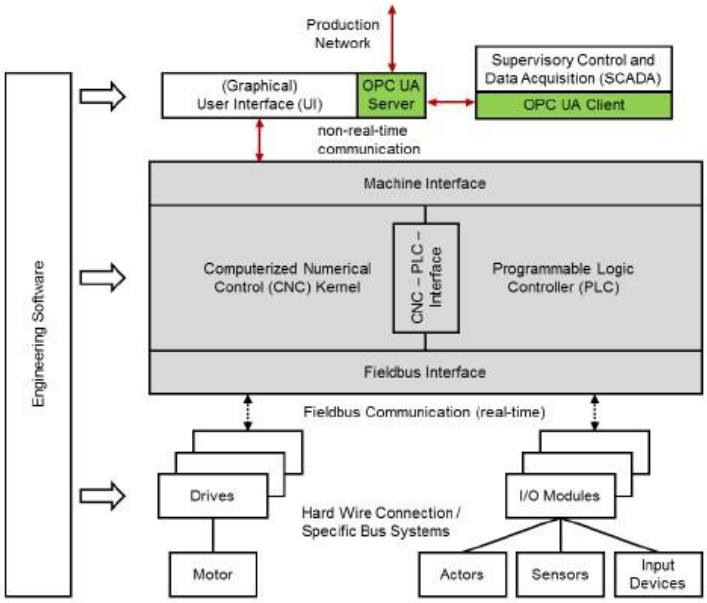


Figura 16 – Arquitetura de sistema desacoplado OPC UA (VDW & OPC Foundation, 2017).

4. Desenvolvimento de servidor OPC UA

4.1. Introdução

O objetivo deste trabalho consistiu na adaptação de uma máquina CNC, existente no laboratório de robótica da ESTG do Politécnico de Leiria, ao paradigma indústria 4.0. Esta máquina precisou de recuperação física, a fazer fora do âmbito do presente trabalho, pelo que já havia sido escolhida a tecnologia para o fazer. Neste caso, a escolha recaiu sobre os produtos da empresa Eding CNC: a placa de controlo (CPU5A3) e o software (Versão: 4.03.32). Pelas razões já aduzidas anteriormente aquando da explicação das características do protocolo, o OPC UA surge como uma boa solução para a implementação da consola de administração para este ativo em particular.

Para a obtenção do servidor foi necessário escolher uma linguagem de programação e a respetiva plataforma de desenvolvimento. O mercado do desenvolvimento de aplicações OPC UA tem várias ofertas, umas privadas e pagas, e outras em projetos de fonte aberta. Considerou-se que, de entre as plataformas de acesso livre e código aberto. seria mais vantajoso optar por utilizar uma biblioteca da fundação OPC, por ser uma plataforma de referência, criada e mantida diretamente por quem desenvolveu e gere a tecnologia OPC-UA. De entre as bibliotecas disponíveis, havia várias opções quanto à linguagem de programação. Atualmente, as *stacks* estão disponíveis em três linguagens diferentes: em .NET, Java e ANSI C. No caso da *stack* em ANSI C a fundação afirma que esta não receberá mais funcionalidades nem atualizações e, por isso, esta opção foi descartada. Após análise, a escolha recaiu sobre a *stack* em .NET tendo-se verificado que estava mais completa, e incluía exemplos e ajudas ao desenvolvimento na forma de tutoriais de utilização, ainda que se tenha também concluído que a documentação é algo limitada. Na Figura 17 está representado o objetivo do projeto, que consiste em obter um servidor baseado em OPC UA que disponibilize a informação relativa à máquina CNC por intermédio da utilização da biblioteca do software Eding.

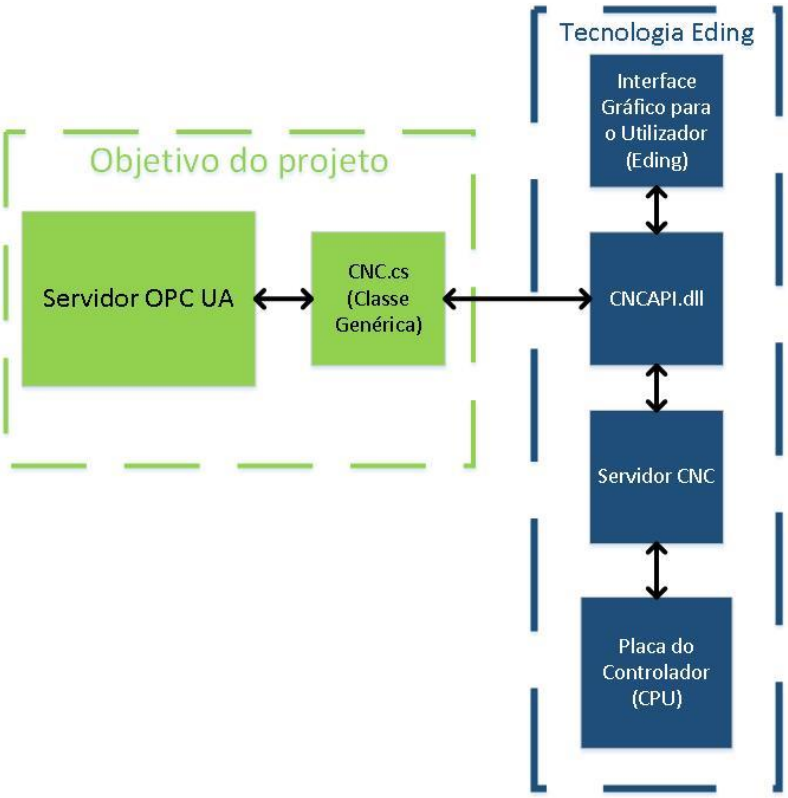


Figura 17 - Diagrama de objetivo do projeto.

4.2. Aplicação sistema CNC

4.2.1. Eding CNC

A empresa Eding CNC é uma empresa holandesa especializada em controlo CNC que tem soluções ao nível de hardware e software para aplicação em variados sistemas CNC.

O software desta empresa permite adaptação e é disponibilizada uma API permitindo o acesso à biblioteca de funções do software e selecionar métodos internos de troca de informação para canalizar a informação relevante de e para o servidor OPC UA.

4.2.2. Software Eding

A interface com o utilizador do software da Eding permite que o comando do sistema CNC possa ser feito a partir de um computador.

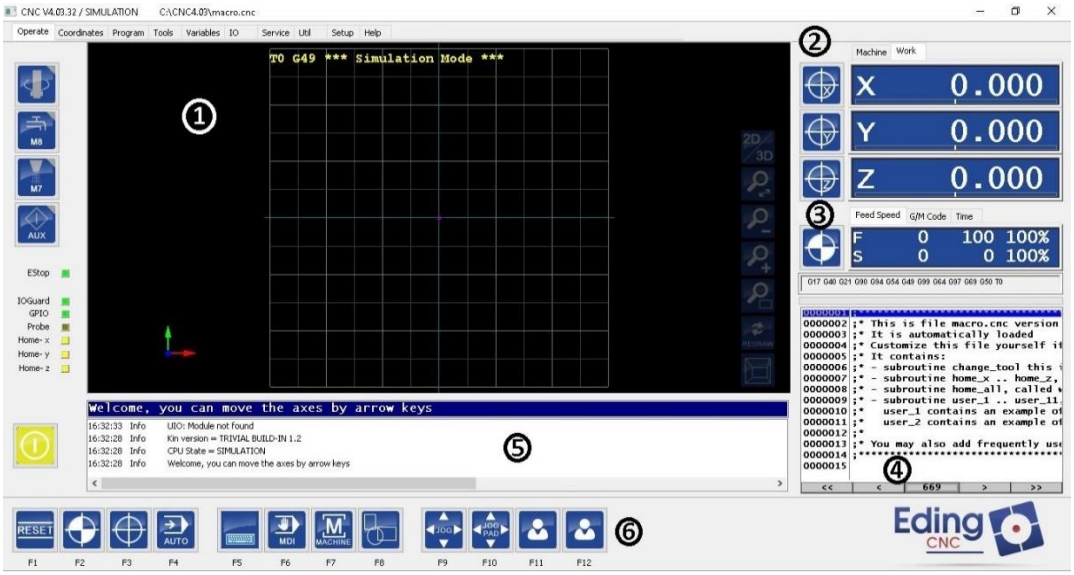


Figura 18 - Interface gráfico do *software* Eding.

Na Figura 18 pode ver-se uma imagem do ecrã, dividindo em diversas zonas, com o seguinte significado:

1- Janela de representação das trajetórias da máquina;
2- Posição dos eixos X, Y e Z, da máquina;
3- Avanço e velocidade de rotação;
4- Janela onde é mostrado o código de trabalho;
5- Janela de mensagens de máquina;
6- Menus de comandos.

O programa foi desenvolvido pela empresa Eding em C++, e utiliza uma biblioteca denominada por 'CNCAPI', que faz a ligação entre a interface do utilizador e o software de comando, o servidor CNC. Este, por sua vez, comunica os comandos à placa controladora que faz o controlo dos motores e da trajetória, conforme podemos ver na Figura 19, que contém o diagrama representativo do funcionamento da tecnologia.

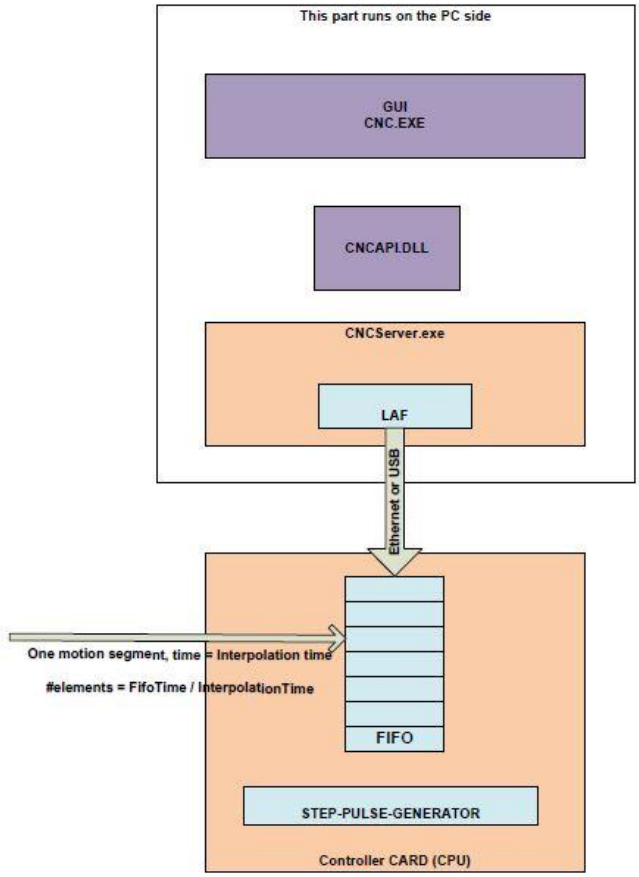


Figura 19 - Esquema de funcionamento da tecnologia Eding (Eding, 2018).

No caso presente, optou-se por criar um programa alojado no servidor OPC UA, para aceder diretamente ao servidor CNC e recuperar/introduzir a informação em paralelo com o interface da Eding, mantendo este para o comando manual da máquina, se necessário, conforme ilustrado na Figura 20 .

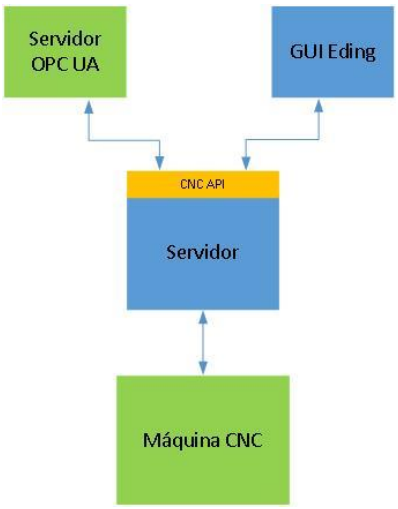


Figura 20 - Interação dos componentes.

Como a linguagem utilizada para o desenvolvimento do servidor é o C# e o software da Eding é em C++, foi preciso recorrer a uma solução para resolver os problemas de compatibilidade. Encontrou-se uma solução desenvolvida por Sander Oosterhof[1] que permite a utilização da biblioteca 'CNCAPI' em C#. Esta solução está disponível na plataforma de partilha de código do .NET, o NuGet. Daqui instalou-se o pacote no projeto permitindo aceder às funções da biblioteca pretendida. Este pacote é gerido pelo programador que a criou.

4.2.3. Classe genérica

O desenvolvimento deste servidor integra-se num projeto mais vasto, cujos objetivos incluem o desenvolvimento de servidores OPC-UA para outros controladores. Nesse âmbito, e em trabalho conjunto[2], foi desenvolvida uma classe C# para modelar um controlador CNC genérico. Em sequência, no presente trabalho, foi desenvolvida a classe derivada que implementa a ligação ao controlador Eding. Deste modo, com o mesmo servidor base, é possível com alterações mínimas, obter servidores para diferentes controladores, já que a implementação específica de cada marca ou modelo é encapsulada numa classe derivada específica. A estrutura desta classe genérica está organizada de uma forma simples: há uma definição abstrata dos métodos que esta irá conter, deixando-se a implementação do método específico para a classe derivada consoante o controlador. A esta classe chamou-se "CNCBase" e a construção desta classe é ilustrada na Figura 21. Na classe abstrata CNCBase são definidos os métodos, também estes abstratos. Por sua vez, na classe derivada, cada uma das definições anteriores será implementada usando os métodos específicos do controlador. Decidiu-se construir a classe genérica desta forma, tanto para permitir a inclusão de outros controladores, como para alargar o número de métodos possíveis. Para uma primeira iteração, optou-se por manter uma definição simples, a que cada método genérico corresponda uma operação específica. Neste caso, cada um destes atualiza um único nó. Os detalhes desta classe estão descritos no anexo A.

[1] https://www.oosterhof-design.com/cncapi-netframework/

[2] Desenvolvido conjuntamente com André Martins no âmbito da sua dissertação de mestrado e do projeto mobilizador Tooling 4G

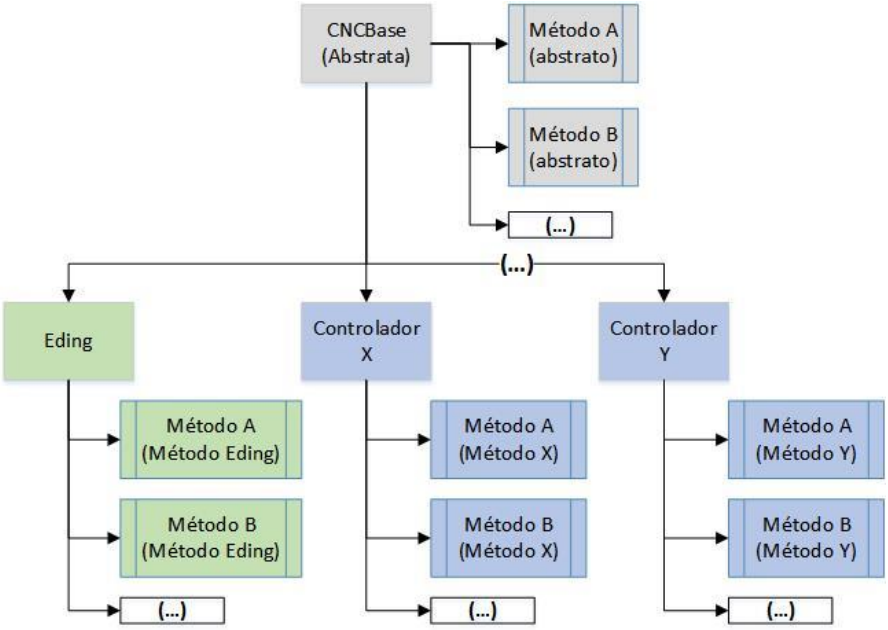


Figura 21 - Esquema de construção da classe genérica.

4.3. Aplicação OPC UA

4.3.1. Introdução

A *stack* em .NET da fundação está disponível como projeto de fonte aberta, com um modelo de licenciamento que permite a sua utilização com fins educacionais, para testes e protótipos. Esta contém a OPC UA *stack* e vários exemplos de referência.

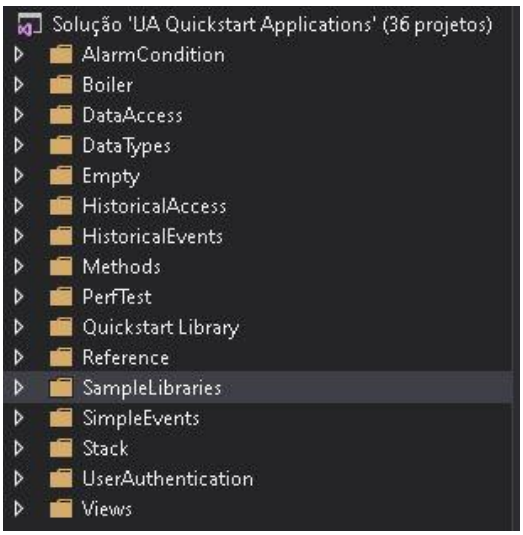


Figura 22 - Árvore de projetos exemplo da *stack* OPC UA.

Na Figura 22 apresentam-se os vários projetos exemplo fornecidos, cada um deles composto por um exemplo de um cliente e respetivo servidor, que permitem simular as várias funções descritas pela norma.

4.3.2. Servidor Base

Um dos exemplos fornecidos pela *stack* da fundação OPC é o tipo *Reference*, exemplo este que foi utilizado durante o desenvolvimento da ferramenta de teste de conformidade (UA CTT) com a norma OPC UA. Na Figura 23 é possível ver o exemplo do aspeto gráfico de um servidor de exemplo da *stack* OPC, neste caso do exemplo *Reference*, onde é mostrado o endereço do servidor, os clientes que se encontram ligados e o número de subscrições ativas.

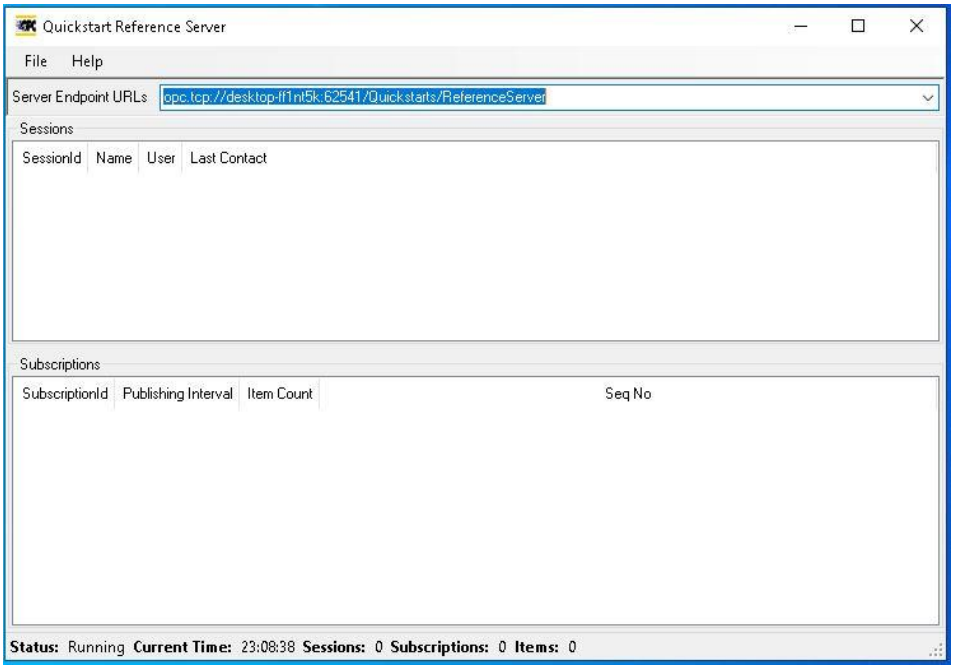


Figura 23 - *Reference Server*.

Na Figura 24 pode ver-se o cliente do mesmo tipo, onde no topo fica identificado o endereço ao qual o cliente se pretende ligar, na janela da esquerda fica a estrutura contida no servidor e, do lado direito, os vários parâmetros de cada nó. Pode ainda ver-se que o *Address Space* é organizado numa estrutura por níveis, dando assim a ideia das dependências dadas.

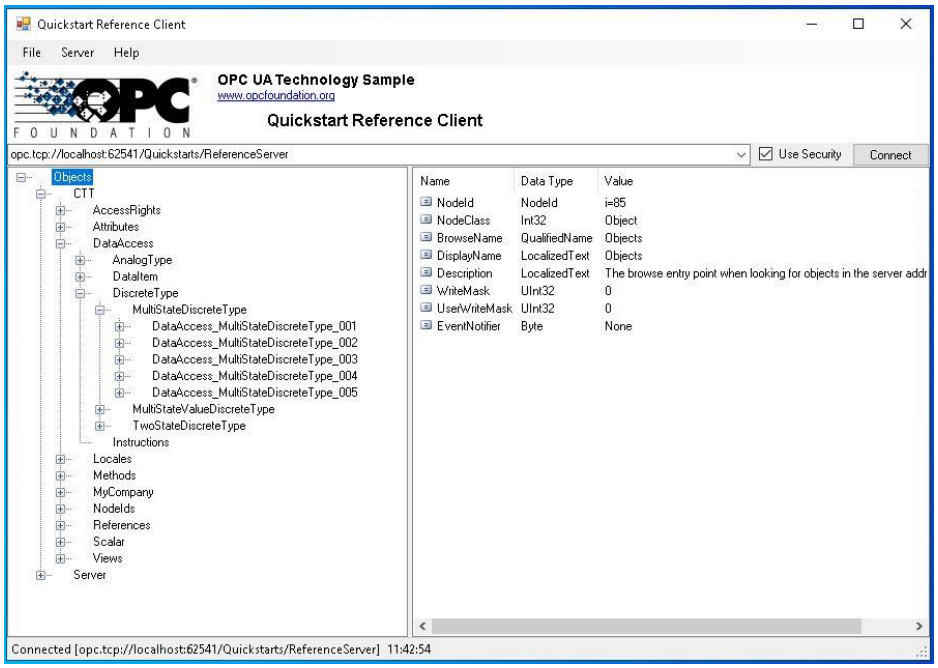


Figura 24 - *Reference Client.*

Escolheu-se este exemplo para basear o servidor principalmente por ser certificado e abrangente, implementando as várias valências da norma. Adicionalmente, pesou também o facto de já existirem tutoriais da fundação sobre como aprender a realizar algumas tarefas ou funções, e, depois, por ter havido alguns desenvolvimentos por parte do professor Luís Perdigoto, da ESTG, que elaborou tutoriais de ajuda ao entendimento do funcionamento de clientes e servidores. O servidor final é, assim, uma adaptação de um projeto *Reference Server* da *stack* original da OPC Foundation.

4.3.3. *Address Space*

O *Address Space* de um servidor OPC UA é o conjunto de objetos e informações de interesse para a aplicação e que são disponibilizadas para o cliente. Na norma, o *Address Space* tem definido como objetivo principal a normalização do conjunto de informação que chega ao cliente, sendo que os objetos podem ser variáveis ou métodos (OPC Foundation, 2017).

A norma OPC UA utiliza o conceito de *namespaces* para criar identificadores únicos entre nomenclaturas de entidades diferentes. Na Figura 25 encontra-se a forma como a *Companion Specification* para sistemas CNC define os *namespaces* a utilizar (VDW & OPC Foundation, 2017).

Namespace	Description	Use
http://opcfoundation.org/UA/	Namespace for *NodeIds* and *BrowsNames* defined in the OPC UA specification. This namespace shall have namespace index 0.	Mandatory
Local Server URI	Namespace for *Nodes* defined in the local *Server*. This may include types and instances used in a device represented by the *Server*. This namespace shall have namespace index 1.	Mandatory
http://opcfoundation.org/UA/CNC/	Namespace for *NodeIds* and *BrowsNames* defined in this specification. The namespace index is *Server* specific.	Mandatory

Figura 25 - *Namespaces* a utilizar num servidor para sistemas CNC (VDW & OPC Foundation, 2017).

Para a obtenção do *Address Space*, seguiram-se as indicações da *Companion Specification*, que, com a definição do modelo de informação, fornece um ficheiro XML formatado de acordo com a Parte 6 da norma. Este contém o modelo completo e permite que se possa processar em computador. A composição da estrutura é feita, neste caso, por três *namespaces*. Associar o conjunto de objetos a um URI permite que eles se diferenciem quando têm o mesmo nome, por exemplo, no caso de serem adicionados conjuntos de objetos que tenham nomes idênticos. O primeiro *namespace* corresponde à norma OPC UA e contém os tipos base dos quais outros 'nós' irão derivar, este terá o índice 0 (zero). O segundo é o *namespace* específico do servidor, com o índice 1, e reúne a estrutura de nós que se pretende associar ao sistema em concreto e o último diz respeito à *Companion Specification,* neste caso a respeitante a equipamentos com um controlador CNC.

4.4. Modelação do *Address Space* para o sistema CNC

A modelação da estrutura foi feita utilizando uma ferramenta comercial, da empresa Unified Automation, o UAModeler®[3]. Esta ferramenta permite gerar o código XML necessário para criar o *Address Space*. Utilizando um interface gráfico, é possível criar a estrutura pretendida de 'nós' e referências obtendo essa mesma estrutura em código, num ficheiro XML para depois se poder adicionar ao servidor para utilização.

A *Companion Specification* fornece um exemplo geral de um modelo de informação para sistemas CNC, mais especificamente, uma máquina de três eixos, como é visível na Figura 26 (VDW & OPC Foundation, 2017). Este foi o exemplo escolhido para ter como referência na modelação do modelo de informação, uma vez que se adequa ao equipamento usado neste projeto.

[3] https://www.unified-automation.com/products/development-tools/uamodeler.html

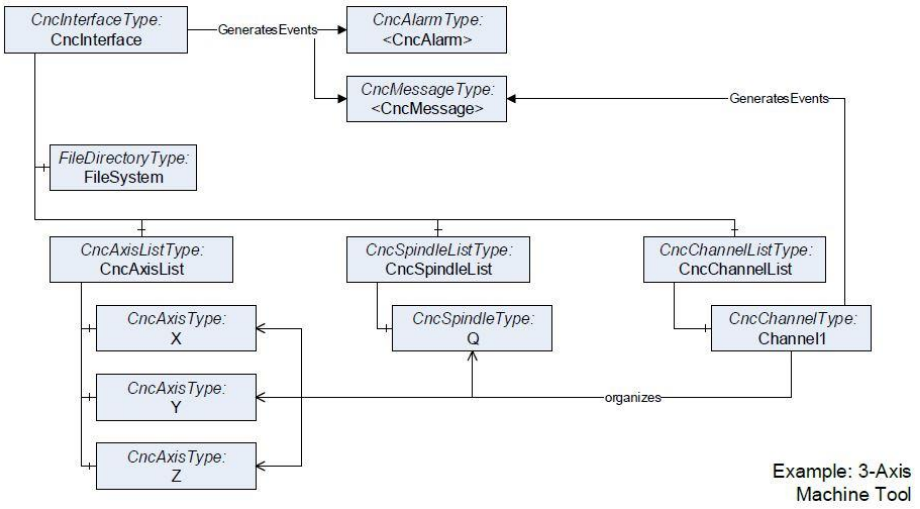


Figura 26 - Exemplo de modelo de informação para sistemas CNC (VDW & OPC Foundation, 2017).

No início de cada projeto, na ferramenta UAModeler® é necessário incluir os *Nodesets* das normas, a incluir no modelo. Para a norma OPC UA, a ferramenta de modelação já inclui um ficheiro na sua biblioteca, pelo que apenas foi necessário acrescentar o *Nodeset* da *Companion Specification*, ficheiro que pode ser obtido através de um endereço disponibilizado pela norma. A Figura 27 mostra o aspeto gráfico do software UAModeler®, em que a janela representada pela letra 'A' representa os ficheiros que foram adicionados ao projeto, a janela com 'B' é o *Address Space* que foi modelado, onde está representada a estrutura e as dependências entre nós, e a janela com 'C' é onde se definem os atributos de cada nó, que incluí as dependências e referências entre nós.

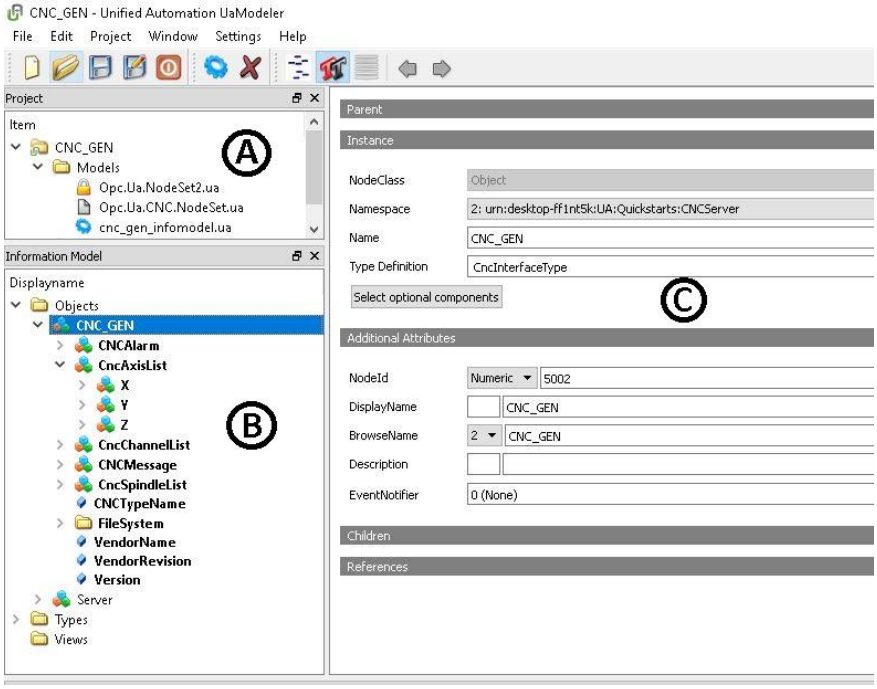


Figura 27 - Aspeto gráfico do software UAModeler.

O *Address Space* utilizado teve em conta dois princípios: a norma da OPC UA e a utilização da classe genérica, já referida anteriormente, podendo ser usada com outros controladores de máquinas CNC. Decidiu-se então que a modelação final seria utilizada para mais do que um sistema e que um dos pontos comuns seriam os identificadores dos nós (*NodeId*).

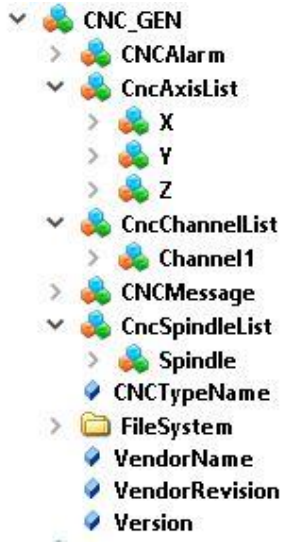


Figura 28 - Estrutura do *AdressSpace*.

A Figura 28 mostra a estrutura gráfica do *Address Space* modelado na ferramenta UAModeler®. Este contém os tipos:

CncInterface: nó responsável pela estruturação da interface. Contém também informações sobre o sistema, como por exemplo a versão;

CncAlarm um nó a ser usado para reportar alarmes;

CncAxisList um nó que deve conter a lista de eixos presentes num sistema. Neste caso, contém os eixos referentes a uma máquina de 3 eixos (X, Y e Z), sendo estes nós do tipo *CncAxis Type*;

CncChannelList um nó que deve conter uma lista de canais que são utilizados pelo sistema, o nó do tipo *CncChannel Type* estará contido na lista e será responsável por administrar os nós do tipo eixo e do tipo *spindle*;

CncMessage é o nó usado para reportar informação simples;

CncSpindleList o nó que contém uma lista de todos os *spindles* afetos ao sistema;

FileSystem é um nó responsável por tratar ficheiros.

4.5. Integração no servidor OPC UA

4.5.1. Atualização do *Address Space*

O *Address Space* é integrado no servidor utilizando métodos da *stack*. Estes métodos partem dos ficheiros XML, onde estão contidos os modelos de informação e o *Address Space* modelado, e carregam a informação para a estrutura, associando a cada um deles um *namespace*. Depois de ser carregado o modelo, precisa de ser atualizado com a informação do equipamento físico, o que, no exemplo de referência é feito através de um método local, sendo repetidamente executado a uma frequência determinada por um temporizador, enquanto o servidor está a correr. Utilizando métodos da stack e incluindo os métodos provenientes da classe genérica, consegue-se fazer uma atualização dos valores dos nós escolhidos. Para a implementação, foi criado um ficheiro acessório XML, onde foi construída uma lista composta por *NodeId* e nomes de métodos da classe genérica, sendo que a cada nó está associado, de acordo com a classe derivada, o método indicado para obter o valor a preencher neste mesmo nó. Um dos parâmetros desta lista é uma entrada binária, sendo o objetivo definir quais os nós a atualizar: caso este valor seja 1 o nó é atualizado, se 0, os métodos que atualizam esse nó não são corridos, o que torna o processo de atualização um pouco mais leve. Nesta fase, os nós que são atualizados correspondem à informação disponibilizada pela biblioteca Eding. Pela forma como foi contruída a classe genérica, em qualquer momento é possível alargar a lista de nós a ser utilizados, através da alteração do ficheiro XML.

4.6. Resultados

Com este trabalho consegue-se um servidor OPC UA que implementa uma biblioteca capaz de funcionar com um controlador de CNC. Para mais detalhes pode-se consultar o manual de utilizador no Anexo A. O servidor em si, está construído para devolver informação sobre o sistema, e ainda não integra métodos para correr ações, gerir eventos nem o registo histórico. Os testes ao funcionamento foram feitos utilizando um cliente baseado em OPC UA da empresa Unified Automation, o UAExpert, que permite a verificação do funcionamento quer da introdução da estrutura, quer da atualização do *Address Space* ao longo do tempo. Não foi possível realizar experiências utilizando a máquina física, uma vez que o projeto paralelo responsável pela recuperação física do equipamento não se concluiu em tempo útil. Porém este projeto não perde generalidade visto que o controlador Eding é o mesmo, estando ou não a máquina fisicamente em operação e o resultado na obtenção do

servidor seria o mesmo, com a diferença de estar a devolver informação sobre o trabalho físico. Na atualização do *Address Space* a opção de se usar a classe genérica facilitou a implementação, pois desta forma é possível selecionar os nós, que se pretendem atualizar e com a frequência de atualização desejada. Ainda não é possível preencher todos os campos do *Address Space* especificado na norma, uma vez que o controlador utilizado não disponibiliza alguns destes dados. Foi, assim, possível obter um servidor baseado em OPC UA utilizando os exemplos da *stack* fornecida pela OPC Foundation.

5. Conclusão e trabalhos futuros

O objetivo principal deste projeto foi alcançado: a utilização da tecnologia OPC UA permite que aplicações de origens diferentes possam comunicar, baseando-se num protocolo comum. Através da normalização de modelos de informação comuns é fácil para os clientes reconhecerem a estrutura e acederem à informação que necessitam. Do ponto de vista da construção de uma consola de administração de ativos, a tecnologia OPC UA parece ser a ideal para fazer face às exigências, pois a norma satisfaz os requisitos. Para a aplicação prática, o caminho foi um pouco mais demorado, pois a documentação é algo reduzida e por ser uma tecnologia relativamente recente, ainda não há muito material partilhado. Foi também interessante perceber que há elasticidade suficiente para introduzir tecnologias, que já possuem o seu mecanismo de comunicação e, fazendo alguns ajustes, colocá-las também a comunicar neste formato. A utilização da classe genérica verifica-se ser uma vantagem, pois é uma forma de intermediar os processos, o que, neste caso, facilitou a implementação por permitir que se pudessem trabalhar os dados antes de os colocar nos nós, o que se tornou necessário nalgumas situações, em que a estrutura de dados, obtida do lado da API da Eding, teve de ser simplificada antes de se poder utilizar a informação.

Como trabalhos futuros fica a implementação do registo histórico do servidor, a gestão de alarmes e eventos e a implementação dos métodos para correr ações no âmbito dos objetos do servidor OPC UA, para que o cliente, para além de poder consultar a informação, possa também interagir diretamente com a máquina. Isto fará com que o servidor se aproxime mais do conceito de consola de administração de ativos, que é o objetivo comum na passagem para os componentes I4.0.

Bibliografia ou Referências Bibliográficas

Adolphs, Peter, et al. 2015. *Reference Architecture Model Industrie 4.0.* Düsseldorf : VDI / VDE & ZVEI, 2015.

Adolphs, Peter, et al. 2016. *Structure of the Administration Shell.* Berlin : Plattform Industrie 4.0 & Federal Ministry of Economic Affairs and Energy (BMWi), 2016.

Eding, Bert. 2018. *Eding CNC user Manual Software.* s.l. : Eding CNC Holding B.V., 2018.

Evans, Dave. 2011. The Internet of Things - How the next evolution of the Internet is changing Everything. [Online] Abril de 2011. [Citação: 03 de 09 de 2019.] https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf.

Gilchrist, Alasdair. 2016. *Industry 4.0: The Industrial Internet of Things.* Bangken, Nonthaburi : Apress, 2016. ISBN: 978-1-4842-2046-7.

Lee, Edward A., et al. Cyber-Physical Systems - a Concept Map. [Online] [Citação: 03 de 09 de 2019.] https://ptolemy.berkeley.edu/projects/cps/.

Lee, Edward Ashford e Seshia, Sanjit Arunkumar. 2011. *Introduction to Embedded Systems - A Cyber-Physical Systems Approach.* s.l. : LeeSeshia.org, 2011. ISBN: 978-0-557-70857-4.

OPC Foundation. 2006. OPC Foundation - Classic. *OPC Foundation.* [Online] 2006. [Citação: 05 de 09 de 2019.] https://opcfoundation.org/about/opc-technologies/opc-classic/.

—. 2017. *OPC Unified Architecture - Part 1: Overview and Concepts.* 2017.

—. 2017. *OPC Unified Architecture - Part 3: Address Space Model.* 2017.

—. 2017. *OPC Unified Architecture - Part 5: Information Model.* 2017.

—. 2006. UA Companion Specifications - OPC Foundation. *OPC Foundation.* [Online] 2006. [Citação: 15 de 08 de 2019.] https://opcfoundation.org/about/opc-technologies/opc-ua/ua-companion-specifications/.

Postół, PhD. Eng. Mariusz. 2016. *20140301E05_DeploymentInformationModel.* Wólczańska : CAS, 2016.

Schwab, Klaus. 2016. *"The Fourth Industrial Revolution".* Cologny : WEF, 2016. ISBN 978-1-944835-01-9.

VDW & OPC Foundation. 2017. *OPC UA Information Model for CNC Systems - Companion Specification.* 2017.

Anexos

Anexo A

Manual de Utilizador

Anexo A - Servidor OPC UA para sistema CNC

João Luís Fernandes Lucas

Leiria, *setembro* de *2019*

Índice

Índice ..... **viii**

Índice **1**

1. Introdução **4**

2. Procedimentos Iniciais **5**

2.1. Solução QuickStart Applications **5**

2.2. Preparação do servidor base **6**

2.2.1. Cópia do servidor 6

2.2.2. Definições de segurança (Opcional) 7

3. Address Space **9**

3.1. Modelação **9**

3.1.1. Novo projeto 9

3.1.2. Modelar o Address Space 12

3.1.3. Exportar o modelo de informação 14

3.2. Integração no servidor **15**

3.2.1. Adicionar namespaces 16

3.2.2. Carregar NodeSets 17

4. Classe genérica **19**

4.1. Estrutura da classe **19**

4.2. Métodos **19**

4.2.1. Log_in() 19

4.2.2. Log_out() 19

4.2.3. Get_x_act_status(); 19

4.2.4. Get_x_dir_actpos() 20

4.2.5. Get_x_indir_actpos() 20

4.2.6. Get_x_is_inactive(); 20

4.2.7. Get_x_is_referenced(); 20

4.2.8. Get_x_zero_offsets(); 20

4.2.9. Get_y_act_status(); 20

4.2.10. Get_y_dir_actpos() 21

4.2.11. Get_y_indir_actpos() 21

4.2.12. Get_y_is_inactive(); 21

4.2.13. Get_y_is_referenced(); 21

4.2.14. Get_y_zero_offsets(); 21

4.2.15. Get_z_act_status(); 21
4.2.16. Get_z_dir_actpos() 22
4.2.17. Get_z_indir_actpos() 22
4.2.18. Get_z_is_inactive(); 22
4.2.19. Get_z_is_referenced(); 22
4.2.20. Get_z_zero_offsets(); 22
4.2.21. Get_act_feedrate(); 22
4.2.22. Get_act_G_functions(); 23
4.2.23. Get_act_jog_increments(); 23
4.2.24. Get_act_main_program_file(); 23
4.2.25. Get_act_main_program_name(); 23
4.2.26. Get_act_M_functions(); 23
4.2.27. Get_modal_offset_function(); 23
4.2.28. Get_operation_mode(); 24
4.2.29. Get_act_program_block(); 24
4.2.30. Get_act_program_name(); 24
4.2.31. Get_act_program_status(); 24
4.2.32. Get_act_channel_status(); 24
4.2.33. Get_feed_hold(); 24
4.2.34. Get_tool_id(); 25
4.2.35. Get_tcp_bcs_x_act_pos(); 25
4.2.36. Get_tcp_bcs_y_act_pos(); 25
4.2.37. Get_tcp_bcs_z_act_pos(); 25
4.2.38. Get_tcp_wcs_x_act_pos(); 25
4.2.39. Get_tcp_wcs_y_act_pos(); 26
4.2.40. Get_tcp_wcs_z_act_pos(); 26
4.2.41. Get_spindle_act_override(); 26
4.2.42. Get_spindle_act_speed(); 26
4.2.43. Get_spindle_act_status(); 26
4.2.44. Get_spindle_act_turn_direction(); 26

5. EdingCNC 27

5.1. Integração no servidor 27

5.2. Notas de utilização 28

6. Servidor OPC UA 28

6.1. Atualização do Address Space 28
6.1.1. Ficheiro cnc.xml 28
6.1.2. Método CreateAddressSpace 29
6.1.3. Método DoSimulation 30

6.2. Testes ao servidor 31
6.2.1. Ligação com servidor 31

6.2.2. Simulação 32

1. Introdução

Este documento pretende fazer uma descrição dos procedimentos necessários para se obter um servidor OPC UA para um controlador da EDING CNC.

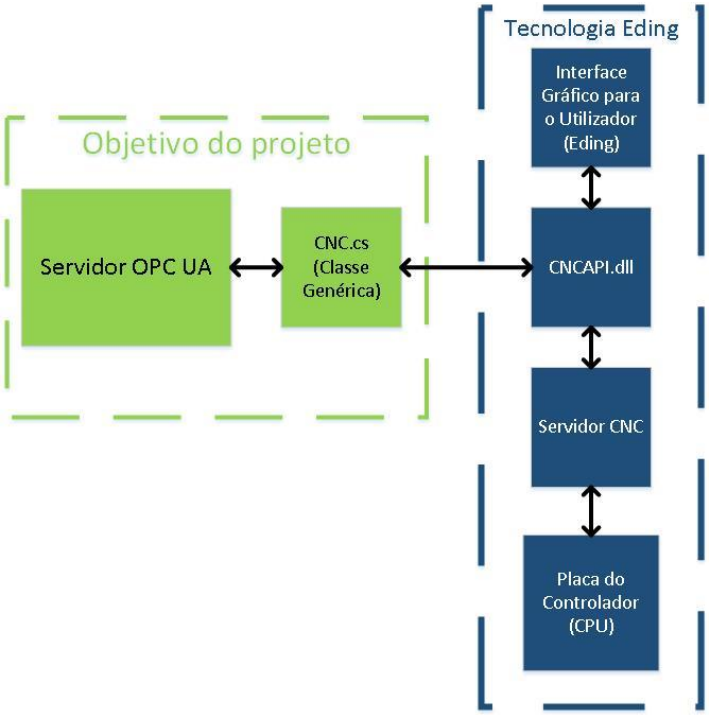


Figura 29 - Objetivo do projeto.

Para o desenvolvimento do servidor os softwares utilizados foram:

-Microsoft Visual Studio Community 2017 (Em anexo está uma lista dos produtos instalados).

- UA Modeler® (Versão 1.6.2-431)

- UA Expert® (Versão 1.5.1-331)

- EdingCNC software (Versão 4.03)

- OPC UA .NETStandard stack and samples master (Versão 1.04.354)

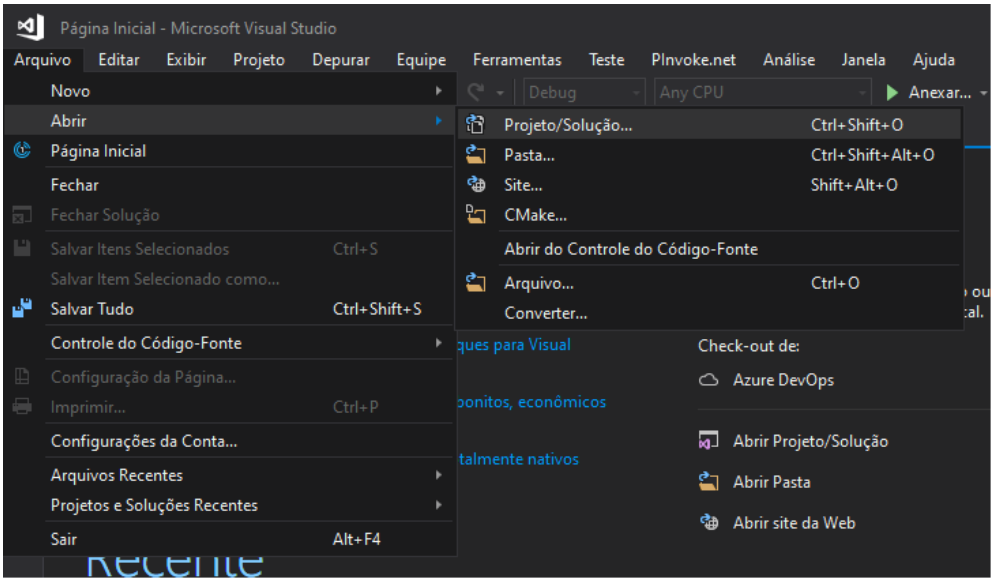
Nota: É possível que os procedimentos apresentados não tenham os mesmos detalhes para outras versões dos softwares aqui indicados.

2. Procedimentos Iniciais

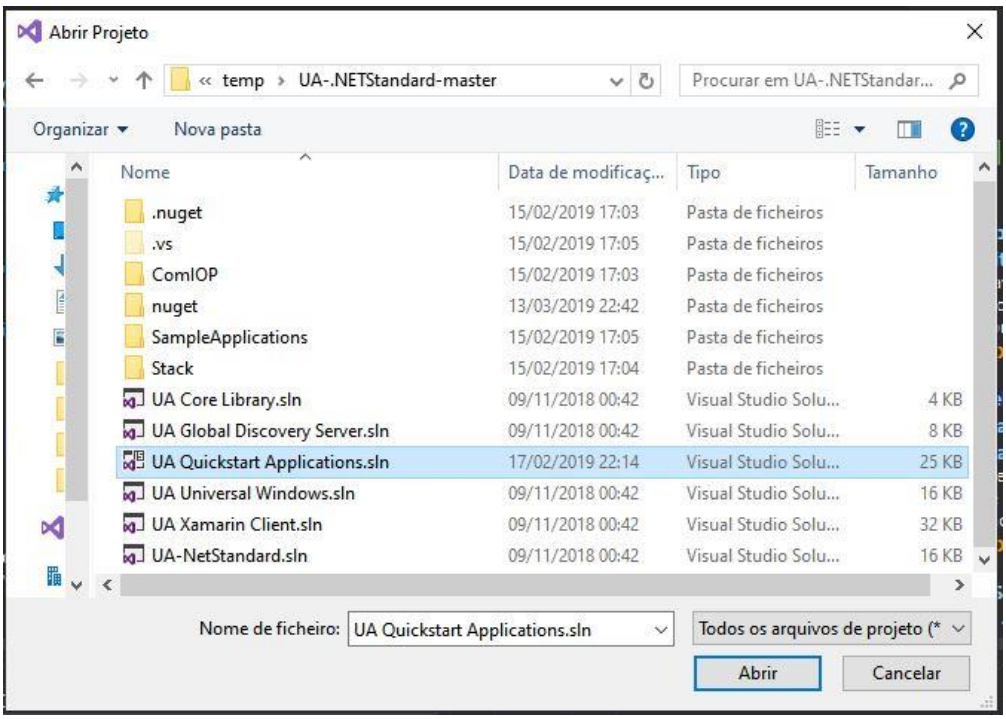
2.1. Solução QuickStart Applications

No Visual Studio:

Arquivo >> Abrir >> Projeto/Solução…



Na diretoria onde está guardada a UA-.NETStandard stack, selecionar a solução “UA Quickstart Applications.sln”.



Para tornar o ambiente de trabalho mais limpo e fácil de aceder, primeiro podem ser removidos os projetos que não se irão utilizar no desenvolvimento do servidor.

Após abrir a primeira vez a stack vê-se uma árvore de projetos, como está na Figura 30 – Transformação da Árvore de projetos da solução Quickstart Applications.Figura 30 (janela da esquerda). A solução pode ter uma árvore mais simples de forma a facilitar a navegação. A janela da direita mostra os projetos utilizados para a construção do servidor.

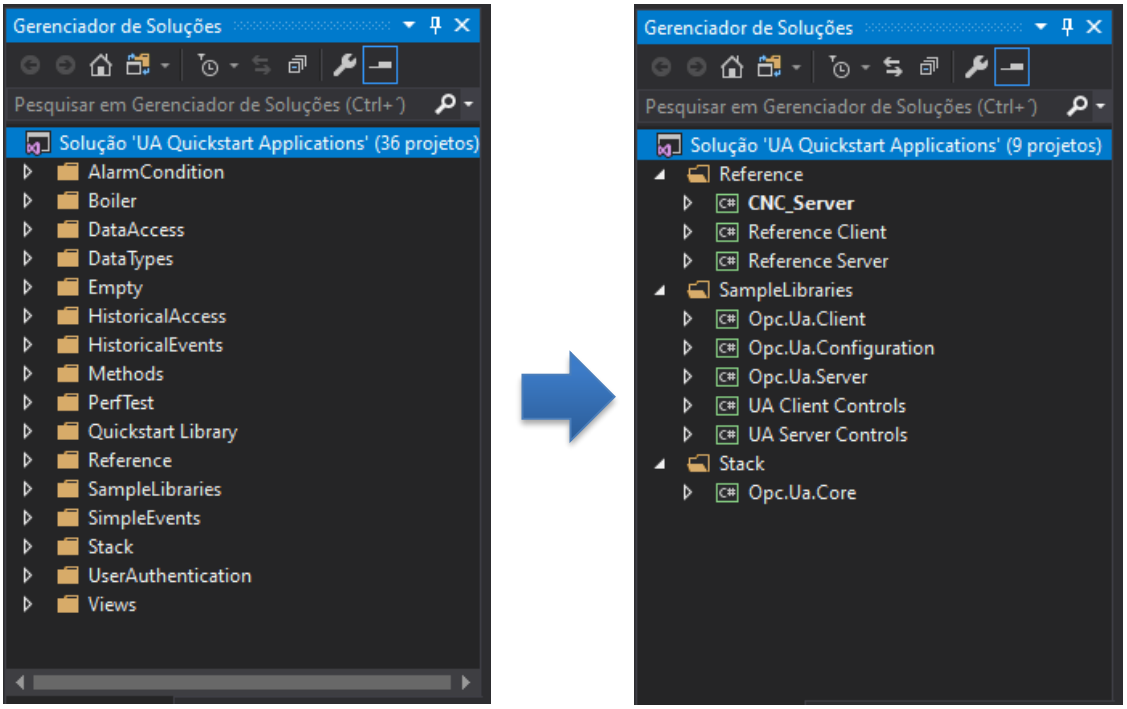


Figura 30 – Transformação da Árvore de projetos da solução Quickstart Applications.

2.2. Preparação do servidor base

2.2.1. Cópia do servidor

O servidor utiliza como base um exemplo da solução, o tipo “Reference Server”. Para este procedimento, faz-se uma cópia do servidor de exemplo:

- Utilizando o Windows Explorer, abrir a diretoria, (…)\UA-.NETStandard-master\SampleApplications\Workshop\Reference;
- Copiar a pasta “Server”, e colar na mesma diretoria;
- Renomear a pasta “Server - Cópia” para o nome pretendido, por exemplo CNCServer;
- Abrir a mesma pasta e renomear o ficheiro "Reference Server.csproj", para o nome pretendido, por exemplo CNCServer.

Após este procedimento adiciona-se o projeto, que se copiou, à solução:

- No Visual Studio, selecionar a pasta “Reference”, com o botão-direito selecionar “Adicionar -> Projeto Existente”;
- Adicionar o ficheiro (nome).csproj renomeado;
- Clicar com o botão direito no projeto adicionado e selecionar “propriedades”;
- Renomear “Nome do assembly” e “Namespace Padrão”, ver Figura 31 para referência;
- Utilizando a ferramenta de pesquisa do Visual Studio, realizar uma pesquisa e substituição de “Reference Server” para o nome pretendido, por exemplo CNCServer.
- Renomear o ficheiro "Quickstarts.ReferenceServer.Config.Xml", para “(nome).Config.Xml”, por exemplo “CNCServer.Config.Xml”.

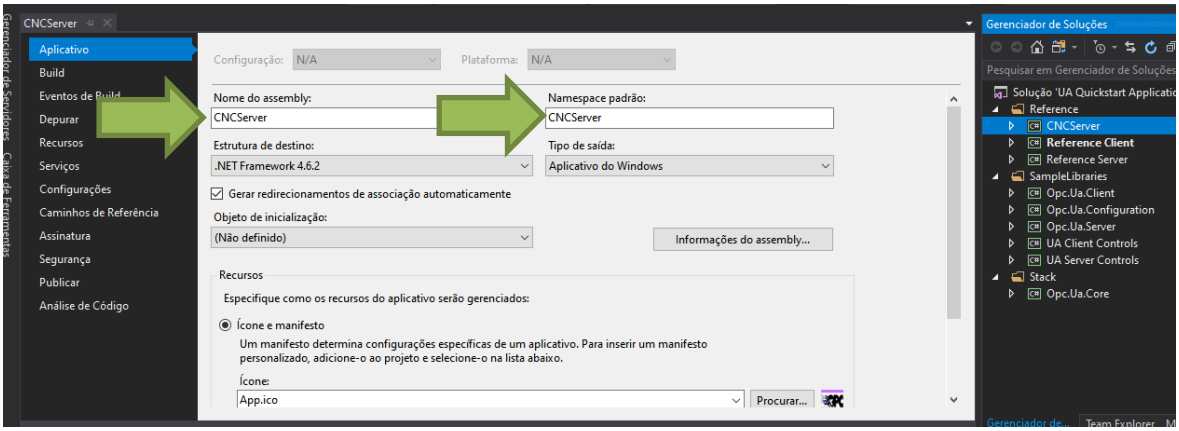


Figura 31 - Propriedades do projeto adicionado.

2.2.2. Definições de segurança (Opcional)

Para alterar as definições de segurança do servidor é necessário abrir o ficheiro “(nome).Config.xml”. Neste ficheiro estão contidas as definições de segurança. Podem lá ser encontradas, as diretorias para guardar os certificados, alterar o modo de ligação de um cliente a um servidor.

```
<SecurityConfiguration>

  <!-- Where the application instance certificate is stored-->
  <ApplicationCertificate>
    <StoreType>Directory</StoreType>
    <StorePath>%CommonApplicationData%\CNCServer\pki\own</StorePath>
    <SubjectName>CN=CNCServer, C=PT, S=Leiria, O=ESTG IPLeiria, DC=localhost</SubjectName>
  </ApplicationCertificate>

  <!-- Where the issuer certificate are stored (certificate authorities) -->
  <TrustedIssuerCertificates>
    <StoreType>Directory</StoreType>
    <StorePath>%CommonApplicationData%\CNCServer\pki\issuer</StorePath>
  </TrustedIssuerCertificates>

  <!-- Where the trust list is stored -->
  <TrustedPeerCertificates>
    <StoreType>Directory</StoreType>
    <StorePath>%CommonApplicationData%\CNCServer\pki\trusted</StorePath>
  </TrustedPeerCertificates>

  <!-- The directory used to store invalid certficates for later review by the administrator. -->
  <RejectedCertificateStore>
    <StoreType>Directory</StoreType>
    <StorePath>%CommonApplicationData%\CNCServer\pki\rejected</StorePath>
  </RejectedCertificateStore>
```

Figura 32 - Diretorias de certificados.

Exemplo de ativação de segurança do servidor.

No ficheiro procurar por “SecurityPolicies”, comentar todos os modos de segurança e deixar só um ativo, o “SignAndEncrypt_3”, como indicado na Figura 33 .

```
<SecurityPolicies>
  <!--<ServerSecurityPolicy>
    <SecurityMode>Sign_2</SecurityMode>
    <SecurityPolicyUri>http://opcfoundation.org/UA/SecurityPolicy#Basic256Sha256</SecurityPolicyUri>
  </ServerSecurityPolicy>
  <ServerSecurityPolicy>
    <SecurityMode>None_1</SecurityMode>
    <SecurityPolicyUri>http://opcfoundation.org/UA/SecurityPolicy#None</SecurityPolicyUri>
  </ServerSecurityPolicy> -->
  <ServerSecurityPolicy>
    <SecurityMode>SignAndEncrypt_3</SecurityMode>
    <SecurityPolicyUri>http://opcfoundation.org/UA/SecurityPolicy#Basic256Sha256</SecurityPolicyUri>
  </ServerSecurityPolicy>
  <!-- deprecated security policies for reference only
  <ServerSecurityPolicy>
    <SecurityMode>Sign_2</SecurityMode>
    <SecurityPolicyUri>http://opcfoundation.org/UA/SecurityPolicy#Basic256</SecurityPolicyUri>
  </ServerSecurityPolicy>
  <ServerSecurityPolicy>
    <SecurityMode>SignAndEncrypt_3</SecurityMode>
    <SecurityPolicyUri>http://opcfoundation.org/UA/SecurityPolicy#Basic256</SecurityPolicyUri>
  </ServerSecurityPolicy>
  <ServerSecurityPolicy>
    <SecurityMode>Sign_2</SecurityMode>
    <SecurityPolicyUri>http://opcfoundation.org/UA/SecurityPolicy#Basic128Rsa15</SecurityPolicyUri>
  </ServerSecurityPolicy>
  <ServerSecurityPolicy>
```

Figura 33 - Lista de modos de segurança.

Comentar a configuração que permite ligação anónima.

```
<!-- The SDK expects the server to support the same set of user tokens for every endpoint. -->
<UserTokenPolicies>
  <!-- Allows anonymous users -->
  <!--<ua:UserTokenPolicy>
    <ua:TokenType>Anonymous_0</ua:TokenType>
    <ua:SecurityPolicyUri>http://opcfoundation.org/UA/SecurityPolicy#None</ua:SecurityPolicyUri>
  </ua:UserTokenPolicy>-->
```

Figura 34 - Configuração de ligação (servidor).

3. Address Space

Para o Address Space é necessário:

- Companion Specification “OPC UA Information Model for CNC Systems”;

  A norma pode ser descarregada através do link:

  https://opcfoundation.org/developer-tools/specifications-opc-ua-information-models/opc-unified-architecture-for-cnc-systems/

  Nota: É necessário o registo na plataforma da OPC Foundation.

- um ficheiro XML com o modelo de informação para sistemas CNC.

Este ficheiro pode ser descarregado através do link:

http://www.opcfoundation.org/UA/CNC/1.0/Opc.Ua.CNC.NodeSet.xml

3.1. Modelação

Modelação do Address Space utilizando a ferramenta UAModeler. Esta ferramenta pode ser descarregada através do link:

https://www.unified-automation.com/downloads/opc-ua-development/file/download/details/uamodeler-v162.html

Nota: É necessário fazer um registo na plataforma da Unified Automation.

3.1.1. Novo projeto

Começar por criar um novo projeto.

File -> New Project

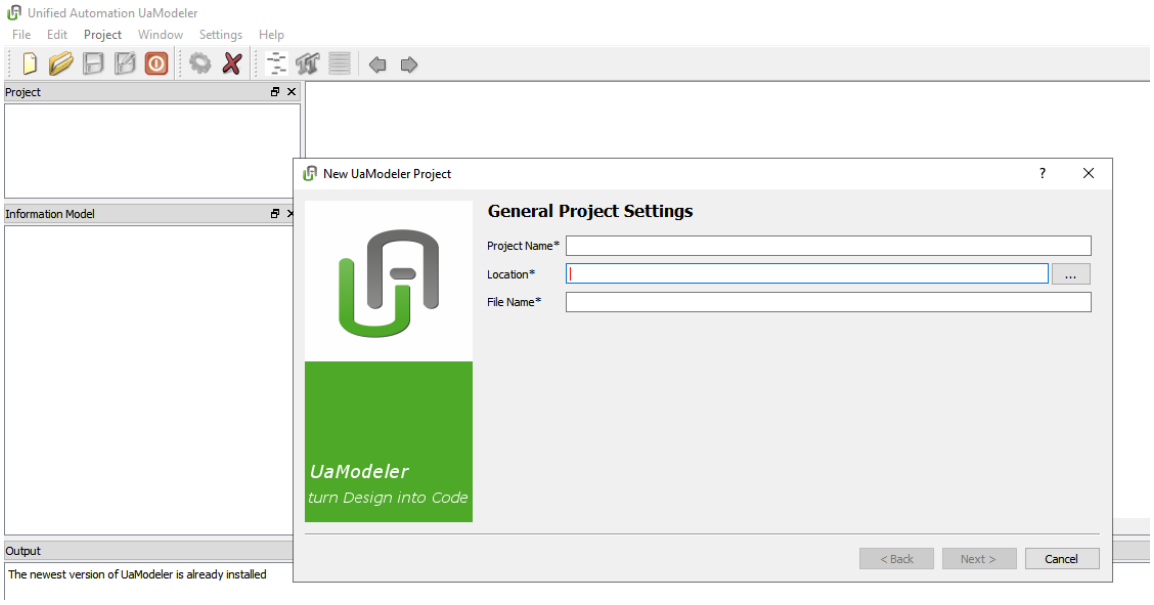


Figura 35 - Novo projeto UAModeler.

Definir o nome do projeto, a diretoria e o nome do ficheiro. O nome colocado no campo "File Name", será o nome do ficheiro XML a utilizar no servidor.

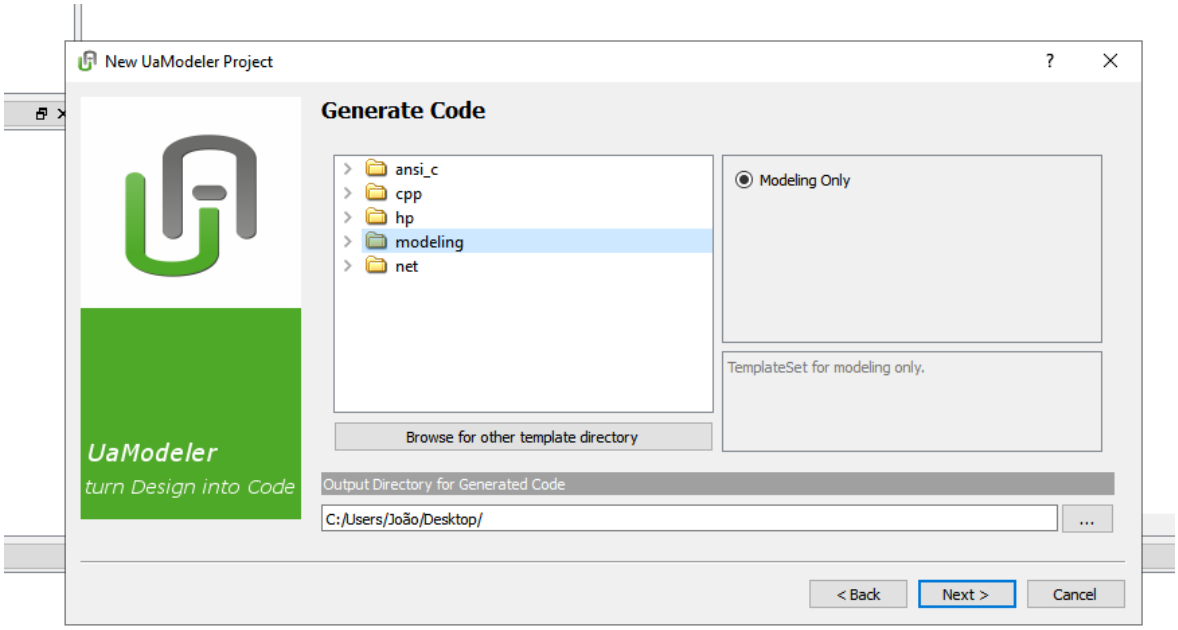


Figura 36 - Novo projeto UAModeler(cont.).

- Selecionar a opção modeling;
- Selecionar a diretoria onde se deseja gravar os ficheiros de saída.

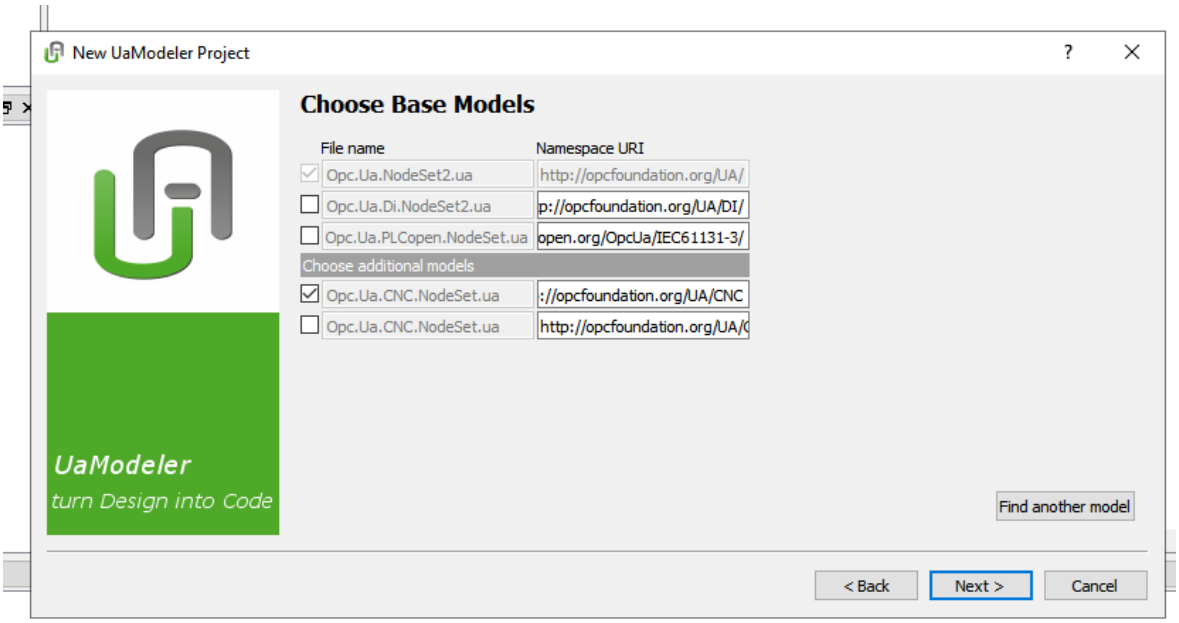


Figura 37 - Novo projeto UAModeler(cont.).

Para adicionar o Nodeset da Companion Specification:

- Selecionar “Find another model”;
- Selecionar o ficheiro Opc.Ua.CNC.NodeSet.xml, previamente adquirido através do link no início do capítulo;
- Selecionar a caixa referente ao NodeSet pretendido;

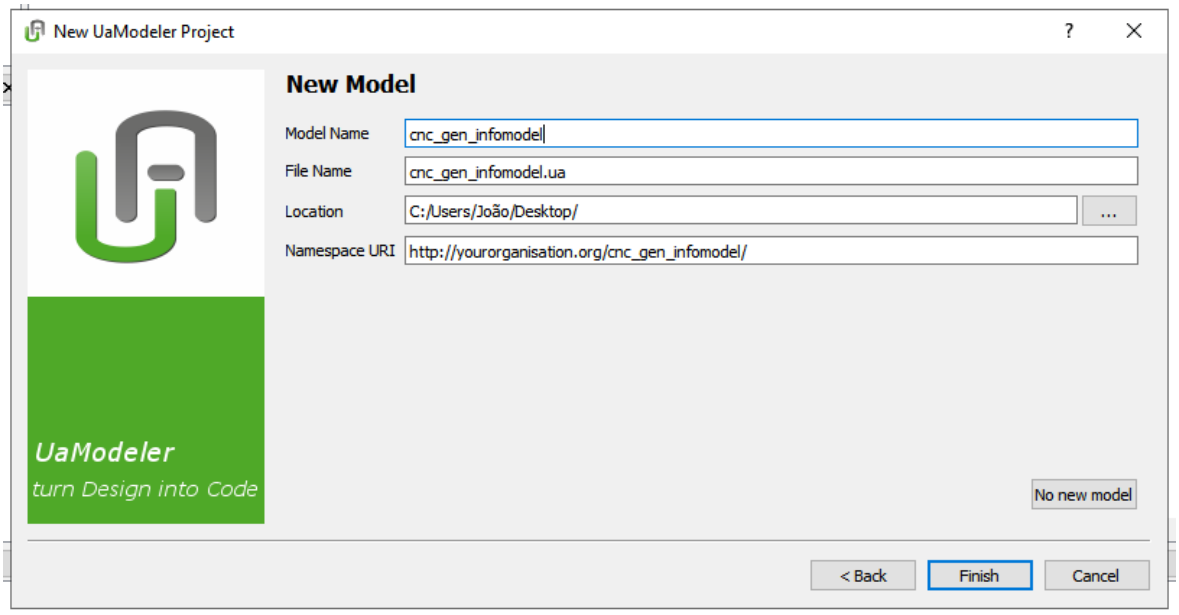


Figura 38 - Novo projeto UAModeler(cont.).

- Introduzir o nome do ficheiro;
- Introduzir endereço namespace, Sendo que este endereço será depois utilizado no servidor.

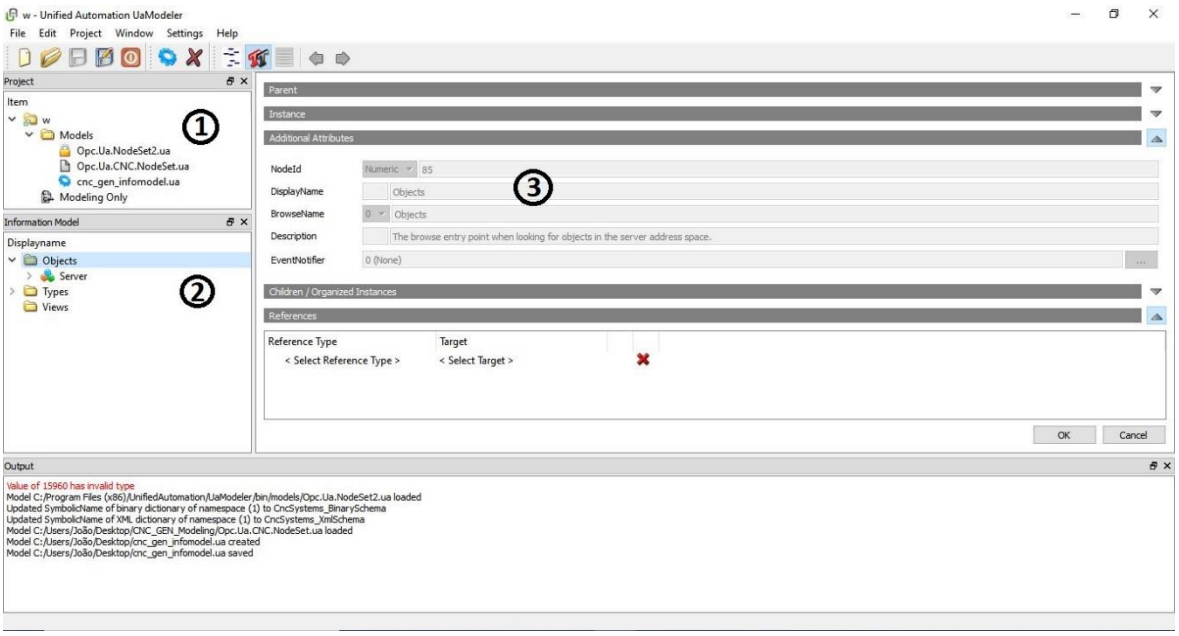


Figura 39 - Novo projeto UAModeler (fim).

O resultado deve ser uma janela semelhante à da Figura 39.
Legenda da Figura 39:
1 – Janela onde encontramos os NodeSets presentes no projeto;
2 – Janela do modelo de informação a modelar;
3 – Janela onde são mostrados os atributos e referências dos nós;

3.1.2. Modelar o Address Space

O objetivo da modelação do Address Space, é alcançar uma estrutura como está representado o exemplo na Figura 40.

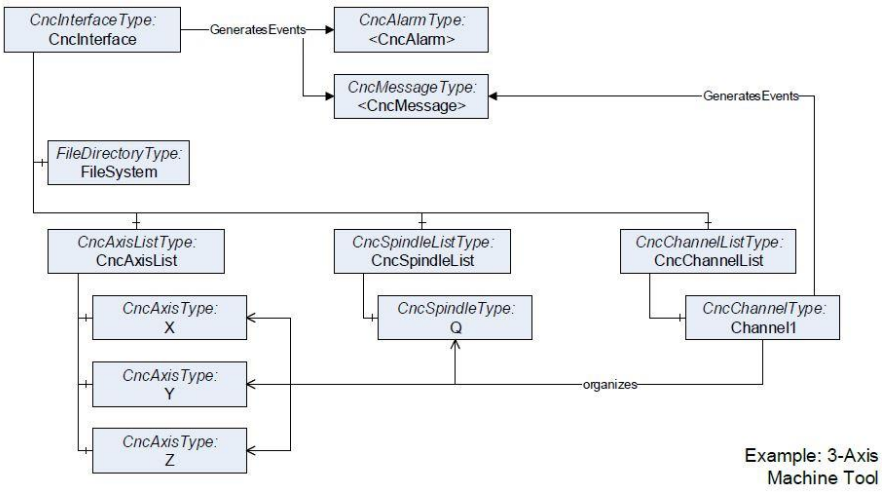


Figura 40 – Exemplo de modelo de informação para sistemas CNC[4].

[4] From: VDW & OPC Foundation. 2017. OPC UA Information Model for CNC Systems - Companion Specification. 2017.

Como referência para identificar as janelas, utiliza-se a legenda da Figura 39.

Na janela 2 indicada pela Figura 39, clicar com o botão direito em cima da pasta “Objects”, e selecionar “Add Instance”.

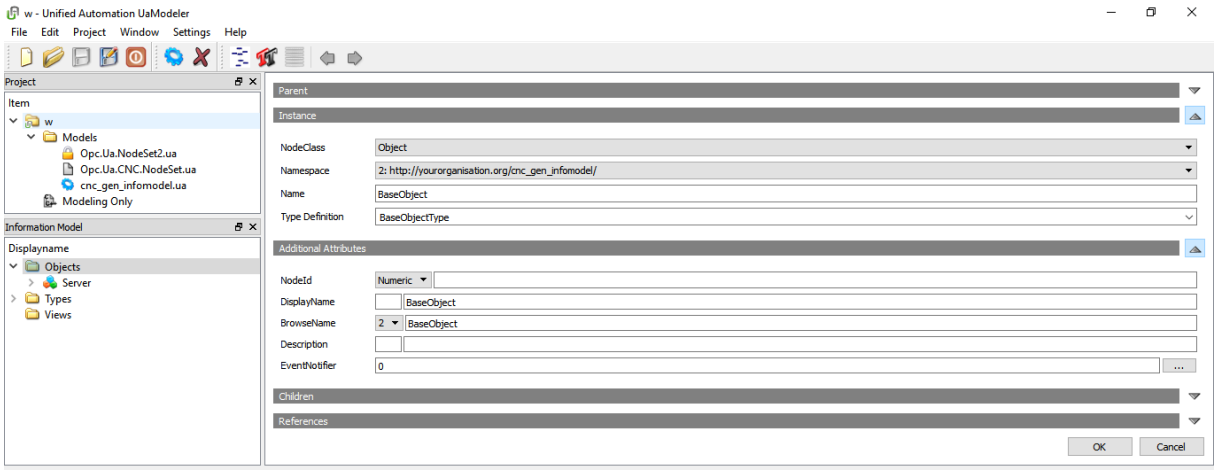


Figura 41 - Adicionar nós.

A diferença que se faz notar, está na janela 3 da Figura 39, onde agora aparece uma lista de parâmetros que podem ser alterados.

Figura 42 - Lista de parâmetros do nó a adicionar.

Indicado na Figura 42, o campo “Name”, serve para dar o nome ao nó, o campo Type Definition serve para definir a classe de nó deste.

Após a escolha do tipo de nó que se está a adicionar, é possível definir os atributos e referências. Na Figura 43, mostra-se como a ferramenta já contempla as regras de modelação das normas.

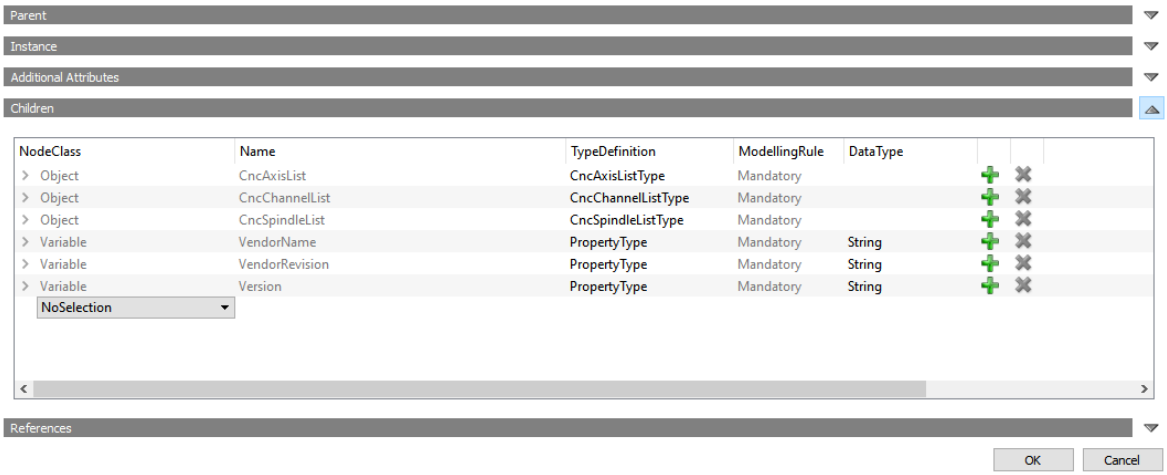


Figura 43 - Adicionar nó (caso dependentes).

3.1.3. Exportar o modelo de informação

Depois de modelado, o Address Space pode ser exportado num ficheiro XML.

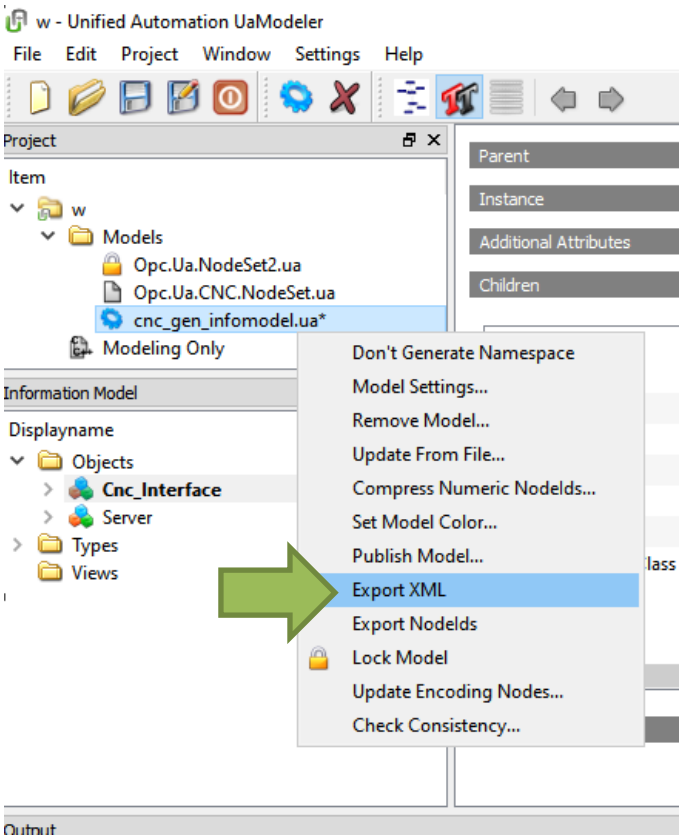


Figura 44 - Exportar Address Space em xml.

Clicar com o botão direito no ficheiro, com o NodeSet, ver na Figura 44, depois escolher "Export XML", como está assinalado na mesma figura.

Nota: Por ser uma licença gratuita, o software tem limite de nós que um modelo pode ter.

3.2. Integração no servidor

Para integrar os modelos de informação no servidor, adicionam-se os ficheiros XML ao projeto destino. Os ficheiros a colocar são, o NodeSet da Companion Specification e o NodeSet do Address Space modelado. É também necessário alterar dois parâmetros nas propriedades do arquivo, como está na Figura 45:

Ação de compilação – Conteúdo;

Copiar para diretório de Saída – Sempre.

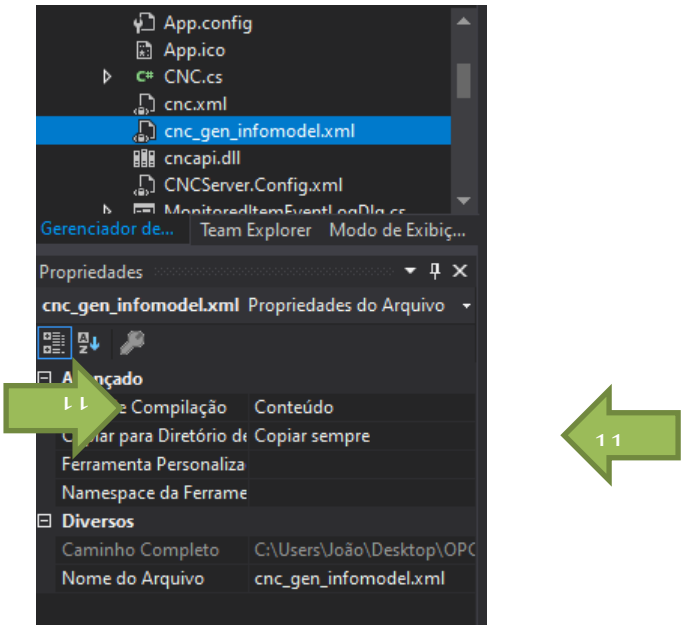


Figura 45 - Propriedades de arquivos adicionados ao projeto.

3.2.1. Adicionar namespaces

Abrir ficheiro ReferenceNodeManager.cs e expandir a região “Constructors”.

```
47          /// Initializes the node manager.
48          /// </summary>
49          public EmptyNodeManager(IServerInternal server, ApplicationConfiguration configuration)
50          :
51              base(server, configuration, Namespaces.ReferenceApplications)
52          {
53              SystemContext.NodeIdFactory = this;
54
55              // get the configuration for the node manager.
56              m_configuration = configuration.ParseExtension<ReferenceServerConfiguration>();
57
58              // use suitable defaults if no configuration exists.
59              if (m_configuration == null)
60              {
61                  m_configuration = new ReferenceServerConfiguration();
62              }
63
64              m_dynamicNodes = new List<BaseDataVariableState>();
65          }
66          #endregion
67
68          IDisposable Members
80
```

Figura 46 - Constructors NodeManager.

Apagar linhas assinaladas na Figura 46 e acrescentar o código abaixo.

```
List<string> namespaceUris = new List<string>();

        // Adicionar namespace para CNC nodeset e adicionar namespace para as variáveis pretendidas
        namespaceUris.Add("http://opcfoundation.org/UA/CNC");
        namespaceUris.Add("urn:desktop-ff1nt5k:UA:Quickstarts:CNCServer");

        NamespaceUris = namespaceUris; //(cont.)
```

```
        m_namespaceIndex =
Server.NamespaceUris.GetIndexOrAppend(namespaceUris[1]);

        var lista = Server.NamespaceUris.ToArray();

        m_lastUsedId = 0;
```

3.2.2. Carregar NodeSets

No ficheiro “ReferenceNodeManager.cs” acrescentar o código abaixo.

```
          #region NodeStateCollection
      protected override NodeStateCollection LoadPredefinedNodes(ISystemContext
context)
      {
         NodeStateCollection predefinedNodes = new NodeStateCollection();
         //Load CNC nodeset
        using (Stream stream = new FileStream("Opc.Ua.CNC.NodeSet.xml",
FileMode.Open)) //NodeSet da Companion Spec.
         {
            try
            {
               Opc.Ua.Export.UANodeSet nodeSet =
Opc.Ua.Export.UANodeSet.Read(stream);
               nodeSet.Import(context, predefinedNodes);
            }
            catch (Exception e)
            {
               Console.WriteLine(e.Message);
            }
         }
         using (Stream stream = new FileStream("cnc_gen_infomodel.xml",
FileMode.Open)) //Address Space modelado
         {
            try
            {
               Opc.Ua.Export.UANodeSet nodeSet =
Opc.Ua.Export.UANodeSet.Read(stream);
               nodeSet.Import(context, predefinedNodes);
            }
            catch (Exception e)
            {
               Console.WriteLine(e.Message);
            }
         }
         return predefinedNodes;
      }
      #endregion
```

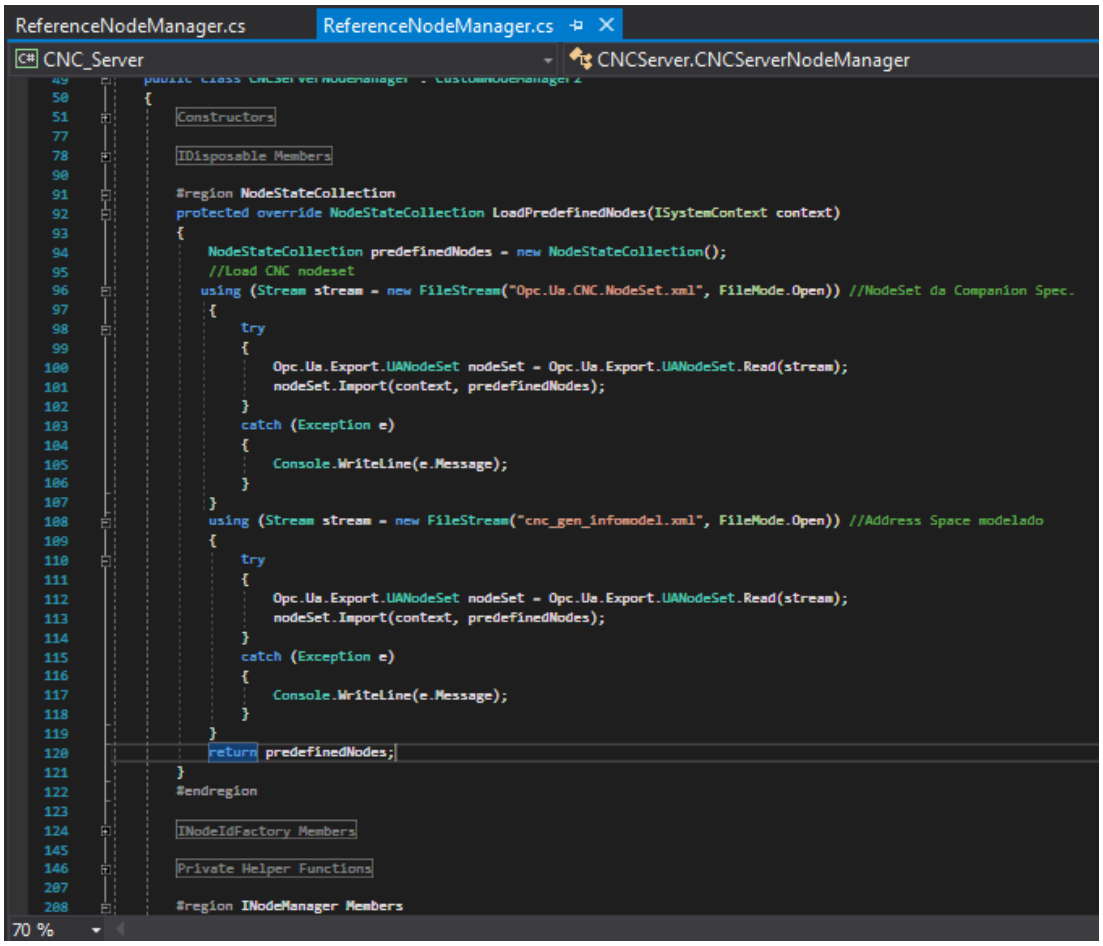


Figura 47 - Exemplo.

Fazer alteração no método “CreateAddressSpace”. Na Figura 48, estão assinaladas as linhas de código que é preciso apagar. Acrescentar a linha: LoadPredefinedNodes(SystemContext, externalReferences);

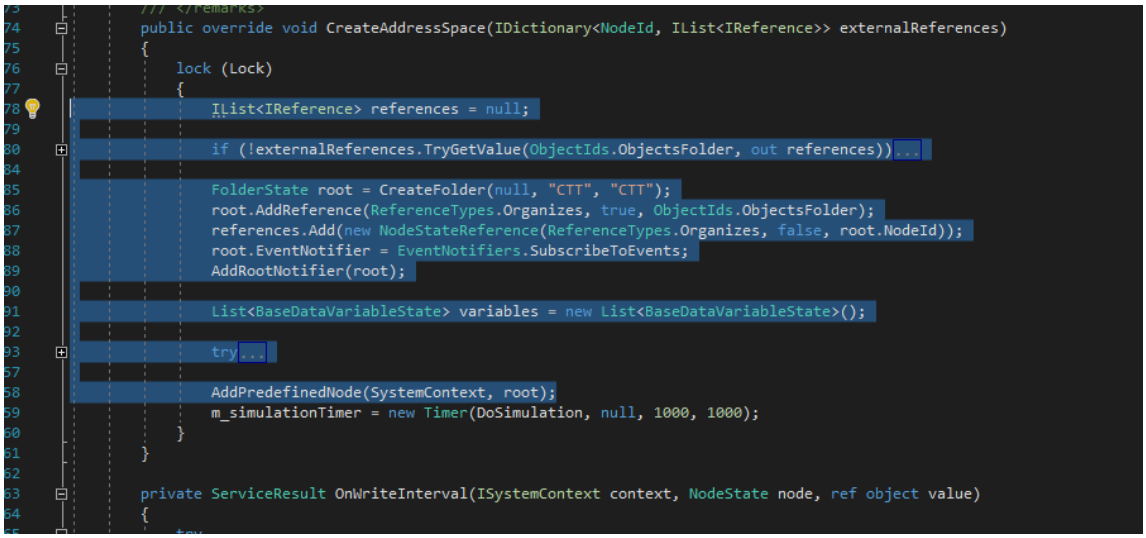
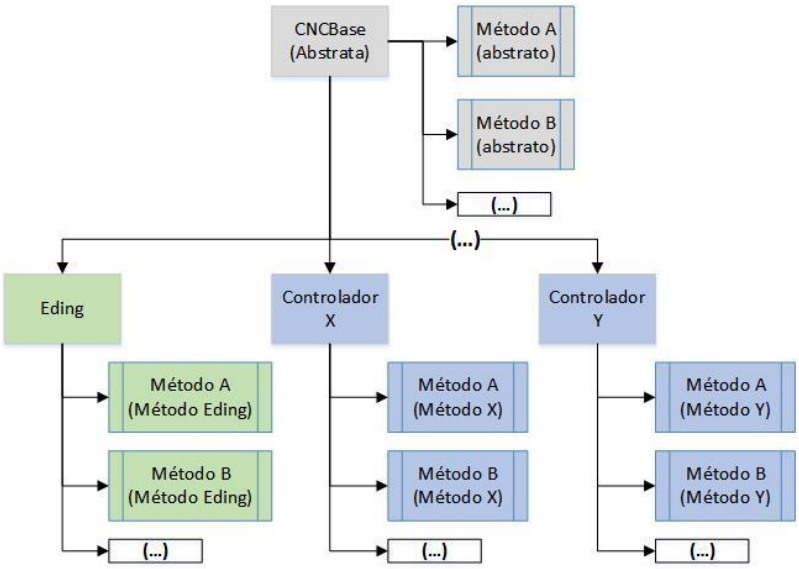


Figura 48 - Método CreateAddressSpace.

Depois de implementar estas alterações no código, já se pode testar o acesso ao address space através do cliente.

4. Classe genérica

4.1. Estrutura da classe



4.2. Métodos

Neste capítulo descrevem-se os métodos para a classe derivada Eding.

4.2.1. Log_in()

Usado para conectar ao servidor Eding. Pode retornar informação sobre operação (ver enum CNC_RC);

```
public override short Log_in() {

}
```

4.2.2. Log_out()

Usado para desconectar do servidor Eding. Pode retornar informação sobre operação (ver enum CNC_RC);

```
public override short Log_out() {

}
```

4.2.3. Get_x_act_status();

Usado para obter informação sobre o estado do eixo X. Ver enum na Companion Specification.

```
public override int Get_x_act_status() {

}
```

4.2.4. Get_x_dir_actpos()

Este método serve para devolver a posição do eixo X da máquina.

```
public override double Get_x_dir_actpos() {
}
```

4.2.5. Get_x_indir_actpos()

Devolve a posição do eixo X, relativo ao referencial ativo.

```
public override double Get_x_indir_actpos(){

}
```

4.2.6. Get_x_is_inactive();

Usado para obter o estado do eixo X.

```
public override boolean Get_x_is_inactive() {

}
```

4.2.7. Get_x_is_referenced();

Usado para obter o estado de referência do eixo X.

```
public override boolean Get_x_is_referenced(); {

}
```

4.2.8. Get_x_zero_offsets();

Usado para mostrar o offset do zero do eixo X.

```
public override double Get_x_zero_offsets() {

}
```

4.2.9. Get_y_act_status();

Usado para obter informação sobre o estado do eixo Y. Ver enum na Companion Specification.

public override int Get_y_act_status() {

}

4.2.10. Get_y_dir_actpos()

Este método serve para devolver a posição do eixo Y da máquina.

```
public override double Get_y_dir_actpos() {
   }
```

4.2.11. Get_y_indir_actpos()

Devolve a posição do eixo Y, relativo ao referencial ativo.

public override double Get_y_indir_actpos(){

}

4.2.12. Get_y_is_inactive();

Usado para obter o estado do eixo Y.

public override boolean Get_y_is_inactive() {

}

4.2.13. Get_y_is_referenced();

Usado para obter o estado de referência do eixo Y.

public override boolean Get_y_is_referenced(); {

}

4.2.14. Get_y_zero_offsets();

Usado para mostrar o offset do zero do eixo Y.

public override double Get_y_zero_offsets() {

}

4.2.15. Get_z_act_status();

Usado para obter informação sobre o estado do eixo Z. Ver enum na Companion Specification.

public override int Get_z_act_status() {

}

4.2.16. Get_z_dir_actpos()

Este método serve para devolver a posição do eixo Z da máquina.

```
public override double Get_z_dir_actpos() {
}
```

4.2.17. Get_z_indir_actpos()

Devolve a posição do eixo Z, relativo ao referencial ativo.

public override double Get_z_indir_actpos(){

}

4.2.18. Get_z_is_inactive();

Usado para obter o estado do eixo Z.

public override boolean Get_z_is_inactive() {

}

4.2.19. Get_z_is_referenced();

Usado para obter o estado de referência do eixo Z.

public override boolean Get_z_is_referenced(); {

}

4.2.20. Get_z_zero_offsets();

Usado para mostrar o offset do zero do eixo Z.

public override double Get_z_zero_offsets() {

}

4.2.21. Get_act_feedrate();

Usado para obter o avanço atual.

public override double Get_act_feedrate() {

}

4.2.22. Get_act_G_functions();

Usado para obter as funções G ativas.

public override int Get_act_G_functions() {

}

4.2.23. Get_act_jog_increments();

Usado para obter o valor do incremento manual.

public override double Get_act_jog_increments() {

}

4.2.24. Get_act_main_program_file();

Usado para obter a diretoria do programa.

public override string Get_act_main_program_file() {

}

4.2.25. Get_act_main_program_name();

Usado para obter o nome do programa principal.

public override string Get_act_main_program_name() {

}

4.2.26. Get_act_M_functions();

Usado para obter as funções M ativas.

public override int Get_act_M_functions() {

}

4.2.27. Get_modal_offset_function();

Usado para obter eixo relativo ativo.

public override int Get_modal_offset_function() {

}

4.2.28. Get_operation_mode();

Usado para obter modo de operação.

public override int Get_operation_mode() {

}

4.2.29. Get_act_program_block();

Usado para obter trechos de código do programa atual.

public override string Get_act_program_block() {

}

4.2.30. Get_act_program_name();

Usado para obter o nome do programa atual.

public override string Get_act_program_name() {

}

4.2.31. Get_act_program_status();

Usado para obter estado atual do programa. Ver enum na Companion Specification.

public override int Get_act_program_status() {

}

4.2.32. Get_act_channel_status();

Usado para obter estado atual do Canal ativo. Ver enum na Companion Specification.

public override int Get_act_channel_status() {

}

4.2.33. Get_feed_hold();

Usado para saber se o avanço está parado.

public override boolean Get_feed_hold() {

}

4.2.34. Get_tool_id();

Usado para obter a identificação da ferramenta.

public override int Get_tool_id() {

}

4.2.35. Get_tcp_bcs_x_act_pos();

Usado para obter a posição do centro da ferramenta no sistema de coordenadas base, em relação ao eixo X.

public override double Get_tcp_bcs_x_act_pos(); {

}

4.2.36. Get_tcp_bcs_y_act_pos();

Usado para obter a posição do centro da ferramenta no sistema de coordenadas base, em relação ao eixo Y.

public override double Get_tcp_bcs_y_act_pos(); {

}

4.2.37. Get_tcp_bcs_z_act_pos();

Usado para obter a posição do centro da ferramenta no sistema de coordenadas base, em relação ao eixo Z.

public override double Get_tcp_bcs_z_act_pos(); {

}

4.2.38. Get_tcp_wcs_x_act_pos();

Usado para obter a posição do centro da ferramenta no sistema de coordenadas da peça de trabalho, em relação ao eixo X.

public override double Get_tcp_wcs_x_act_pos() {

}

4.2.39. Get_tcp_wcs_y_act_pos();

Usado para obter a posição do centro da ferramenta no sistema de coordenadas da peça de trabalho, em relação ao eixo Y.

public override double Get_tcp_wcs_y_act_pos() {

}

4.2.40. Get_tcp_wcs_z_act_pos();

Usado para obter a posição do centro da ferramenta no sistema de coordenadas da peça de trabalho, em relação ao eixo Z.

public override double Get_tcp_wcs_z_act_pos() {

}

4.2.41. Get_spindle_act_override();

Usado para obter o valor atual do override do spindle.

public override double Get_spindle_act_override() {

}

4.2.42. Get_spindle_act_speed();

Usado para obter a velocidade atual do spindle.

public override double Get_spindle_act_speed() {

}

4.2.43. Get_spindle_act_status();

Usado para obter o estado atual do spindle.

public overrid int Get_spindle_act_status() {

}

4.2.44. Get_spindle_act_turn_direction();

Usado para obter o sentido de rotação do spindle.

public override int Get_spindle_act_turn_direction() {

}

5. EdingCNC

5.1. Integração no servidor

Para a utilização da API do software da Eding, é necessário instalar um pacote NuGet do programador Sander Oosterhof[5]. O método é:

- Abrir o ficheiro “UA Quickstart Applications.sln”;
- No Visual Studio: “Ferramentas” -> ”Gerenciador de Pacotes do NuGet” -> “Gerenciar Pacotes do NuGet para a Solução…” ; (como está na Figura 49)
- Procurar por “Oosterhof” ou “CncApi_Net” e selecionar o projeto em que se pretende instalar o pacote. (como se pode ver na Figura 50)

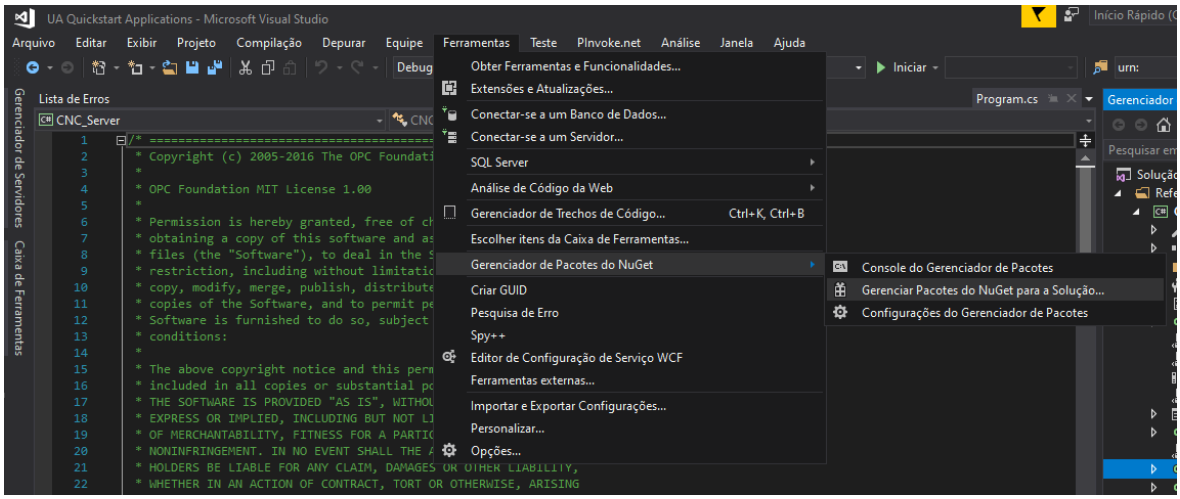


Figura 49 - Abrir NuGet manager.

[5] https://www.oosterhof-design.com/cncapi-netframework/

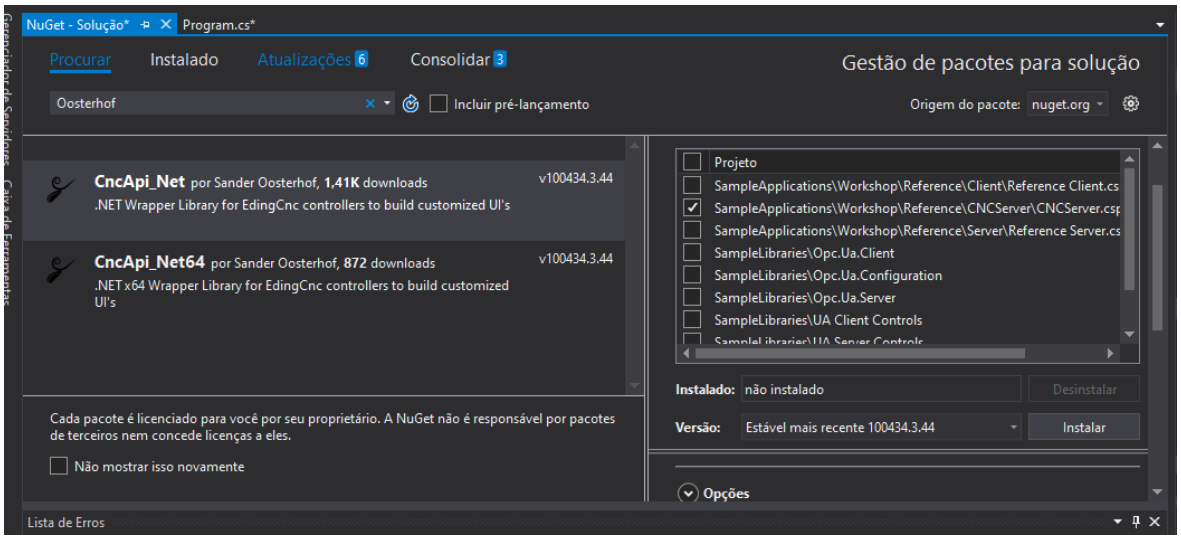


Figura 50 - Instalar pacote CncApi_Net.

5.2. Notas de utilização

Após a instalação só precisa de ser referenciado o namespace, no início do ficheiro através da linha:

"using OosterhofDesign;"

Para utilizar os métodos é necessário consultar as bibliotecas do software EdingCNC, na diretoria "(…)\CNC4.03\cncapi" encontram-se os ficheiros referentes à biblioteca da API da Eding, ver cncapi.h para ter acesso à descrição completa dos métodos. Para utilizar o wrapper de Oosterhof, digite 'G_' seguido do nome da classe a que pretende aceder.

6. Servidor OPC UA

6.1. Atualização do Address Space

6.1.1. Ficheiro cnc.xml

Para a chamada de métodos mais automática da classe, contruiu-se um ficheiro XML com o seguinte formato: (Para este projeto chamou-se cnc.xml)

<add key="nome" value="NodeId,bool atual,int temp"/>

em que:

- nome – Escrever o nome do método da classe;
- NodeId – Número de nó correspondente ao resultado do método, no namespace geral;

- atual – Parâmetro que confirma se a entrada está ativa ou não;
- temp – Valor que serve de entrada para temporizador, em segundos;

Por exemplo, a Figura 51 é um pequeno excerto do ficheiro XML, as entradas do documento assinalam os dois métodos que irão ser chamados para atualizar os valores dos nós. Isto significa que o método "Get_x_dir_actpos" irá atualizar o nó com o identificador '6008' de um em um segundos.

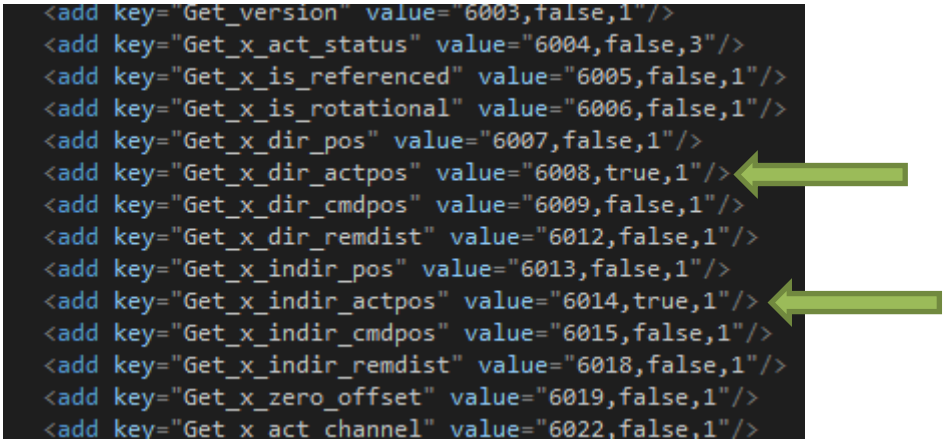


Figura 51- Excerto de ficheiro XML acessório à classe genérica.

6.1.2. Método CreateAddressSpace

Como os mecanismos que se vão apresentar só precisam de correr uma vez no início da aplicação, foram acrescentados no método CreateAddressSpace. No ficheiro ReferenceNodeManager.cs, expandir a região INodeManager Members, e acrescentar o código abaixo.

```
PropertyState modelnode;
   modelnode = (PropertyState)FindPredefinedNode("ns=1;i=6001", null);
   modelnode.ClearChangeMasks(SystemContext, false);
   string modelo = modelnode.Value.ToString();

   if (modelo == "FANUC")
   {
      machine = new CNCFanuc();
   }
   else if (modelo == "HEIDENHAIN")
   {
      machine = new CNCHeidenhain();
   }
   else if (modelo == "EDING")
   {
      machine = new CNCEding();
   }

   Type estetipo = typeof(CNCBase);
```

```
string xml = "cnc.xml";
   XmlDocument xfile = new XmlDocument();
   xfile.Load(xml);
   String res = "";
   XmlNodeList nodeList = xfile.GetElementsByTagName("add");
   //CRIA LISTA COM OS NOS A ATUALIZAR
   foreach (XmlNode node in nodeList)
   {
      var key = node.Attributes["key"].Value;
      if (key.StartsWith("Get"))
      {
         res = node.Attributes["value"].Value;
         string[] StringArray = res.Split(',');
         if (StringArray[1] == "true")
         {
            Variable v = new Variable
            {
               method_name = key,
               nodeid = int.Parse(StringArray[0])
            };
            v.method = estetipo.GetMethod(v.method_name);
            v.period = int.Parse(StringArray[2]);
            methods.Add(v);
         }
      }
   }
```

O que este excerto de código vai fazer é, por ordem:

- Encontrar o nó com a propriedade referente ao fabricante do controlador;
- Gravar o valor numa variável;
- Avaliar a variável;
- Criar um objeto novo do tipo de controlador que tiver recebido como entrada.
- Através do ficheiro cnc.xml vai criar uma lista com os nós que estão ativos para atualização.

6.1.3. Método DoSimulation

O método “DoSimulation” faz parte de um dos métodos exemplos que vem com o servidor da stack. Este método é da classe Timer, do namespace System.Windows.Forms, consiste num temporizador que chama um método de acordo com um intervalo definido.

Dentro do método DoSimulation acrescentar o código:

```
machine.Log_in();
            foreach(Variable method in methods)
            {
```

```
                DateTime Time = DateTime.Now;
                TimeSpan Period = new TimeSpan(0, 0, 0, 0, (method.period * 100));
                if(Time>= method.time)
                {
                    method.time = Time.Add(Period);
                    BaseDataVariableState resultado;
                    resultado =
(BaseDataVariableState)FindPredefinedNode("ns=1;i="+method.nodeid.ToString(),
null);
                    resultado.Value = method.method.Invoke(machine, null);
                    resultado.ClearChangeMasks(SystemContext, false);
                }
            }
```

De modo que o método fique como está na Figura 52.

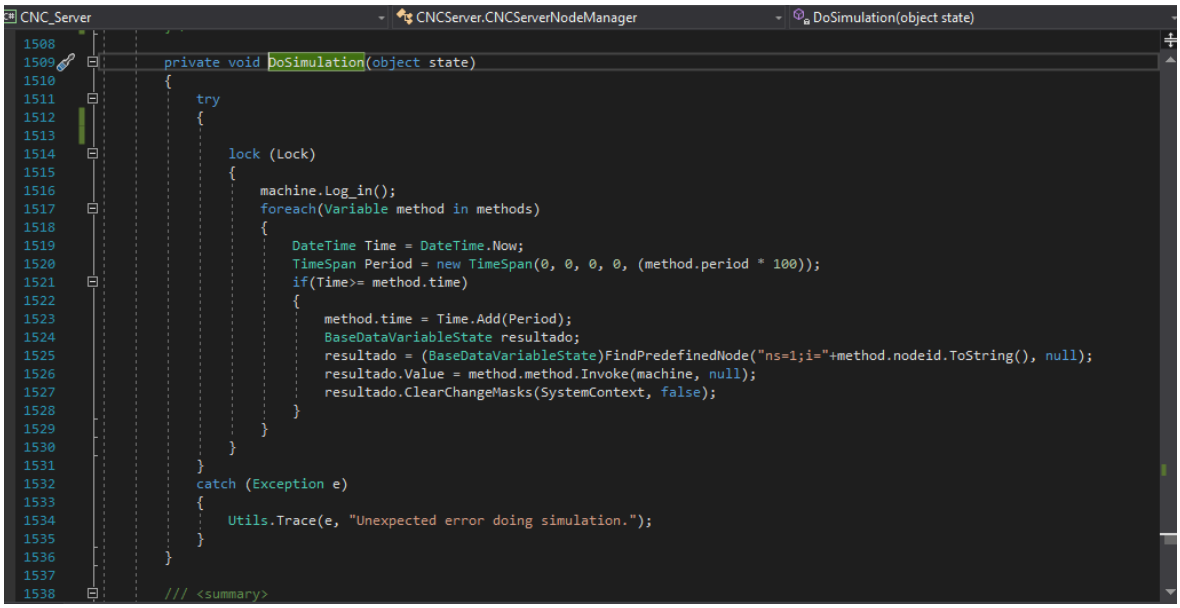


Figura 52 - Método DoSimulation.

6.2. Testes ao servidor

Para testar o servidor foi usado um cliente genérico da empresa Unified Automation.

Utilizou-se o UAExpert para fazer as ligações com o servidor.

6.2.1. Ligação com servidor

No UAExpert:

Server -> Add

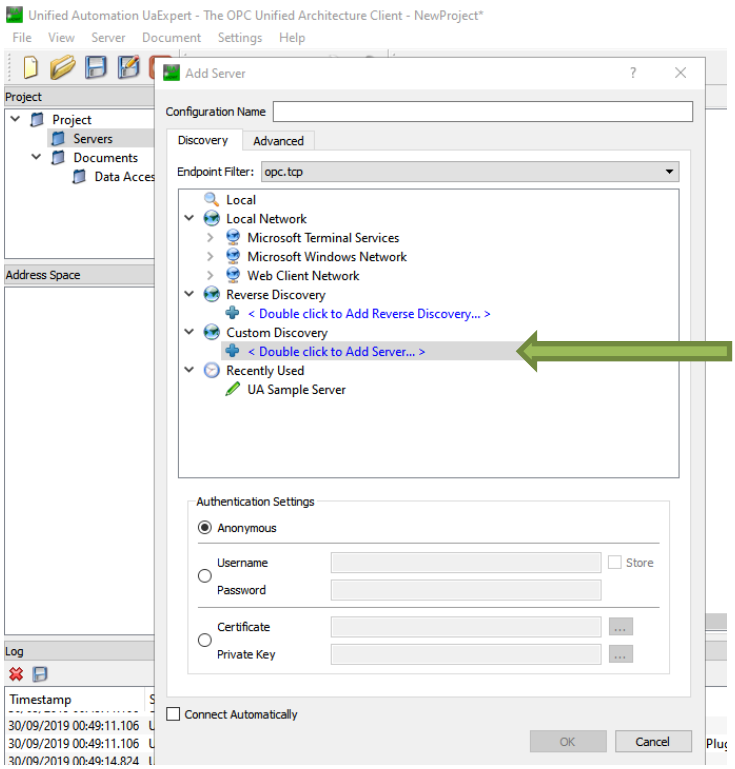


Figura 53 - Adicionar servidor.

- Selecionar a opção indicada na Figura 53;
- Colocar URL do servidor;
- Selecionar o modo de ligação com o servidor
- Por defeito as credenciais para introduzir são: username – "sysadmin" e password – "demo".

6.2.2. Simulação

Na Figura 54 mostra a ligação de um cliente de uma outra fonte a aceder ao servidor CNC.

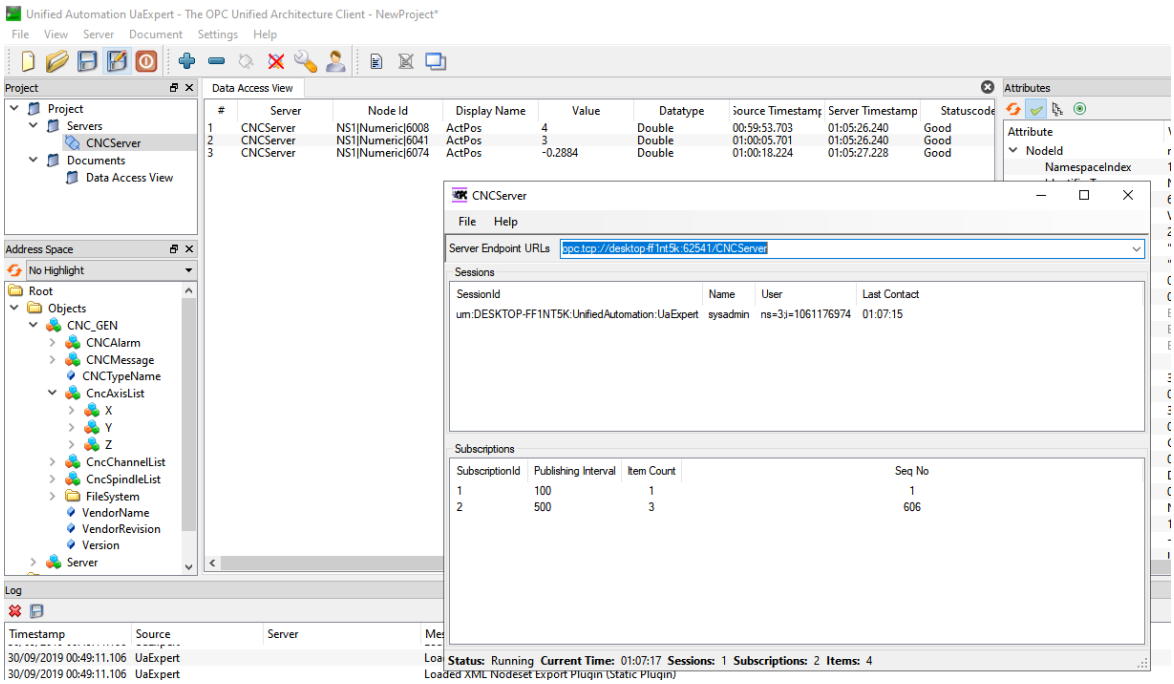
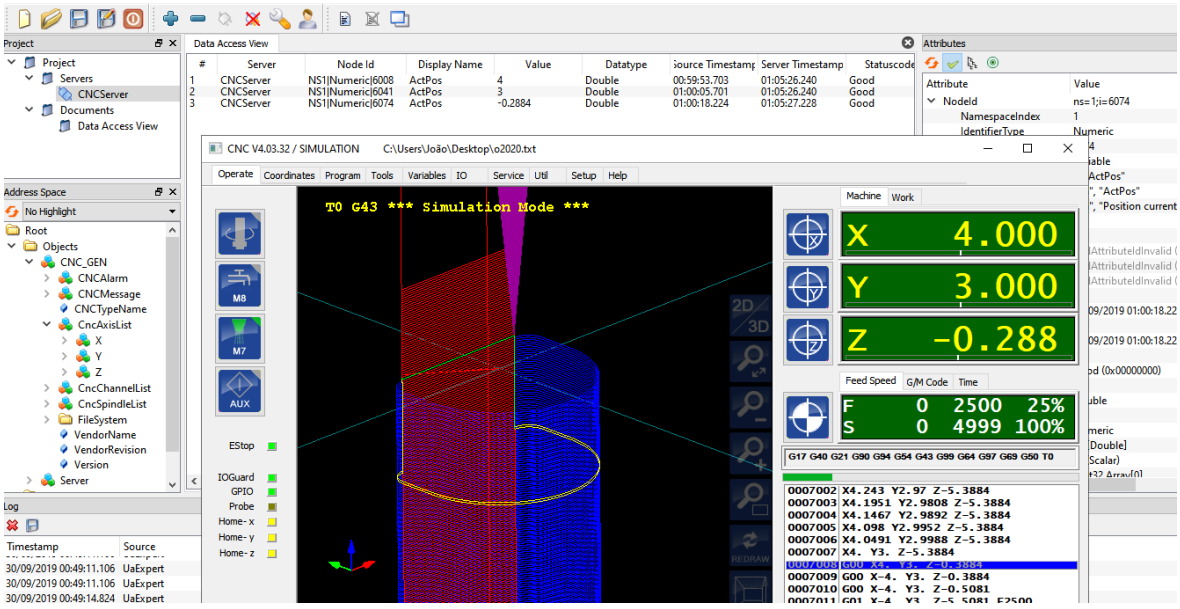


Figura 54 - Ligação com o servidor CNC.

A

1	CNCServer	NS1|Numeric|6008	ActPos	4	Double	00:59:53.703	01:05:26.240	Good
2	CNCServer	NS1|Numeric|6041	ActPos	3	Double	01:00:05.701	01:05:26.240	Good
3	CNCServer	NS1|Numeric|6074	ActPos	-0.2884	Double	01:00:18.224	01:05:27.228	Good

Figura 55 ilustra o software Eding em funcionamento e o cliente recebendo os valores dos nós monitorizados.



1	CNCServer	NS1|Numeric|6008	ActPos	4	Double	00:59:53.703	01:05:26.240	Good
2	CNCServer	NS1|Numeric|6041	ActPos	3	Double	01:00:05.701	01:05:26.240	Good
3	CNCServer	NS1|Numeric|6074	ActPos	-0.2884	Double	01:00:18.224	01:05:27.228	Good

Figura 55 - Cliente ligado com Eding em simulação.