COMPONENTES DE SOFTWARE EM LABVIEW PARA INTEGRAÇÃO EM SISTEMAS DE INFORMAÇÃO INDUSTRIAIS

Manuel M. Carvalho
Unidade de Automação Industrial e Electrónica
IDITE-Minho
Braga

João M. Fernandes
Dep. Informática
Escola de Engenharia
Universidade do Minho
Braga

Ricardo J. Machado
Dep. Sistemas de Informação
Escola de Engenharia
Universidade do Minho
Guimarães

Sumário

O objectivo principal deste artigo consiste em apresentar um conjunto de componentes *LabVIEW*, desenvolvidos com o intuito de facilitar a construção de soluções orientadas ao controlo (ICIS), tendo em conta a respectiva integração e interligação com o sistema de informação de gestão (MIS) duma organização industrial. O artigo discute igualmente os diversos níveis CIM e quais aqueles que são abarcados pelo trabalho apresentado. É também feita uma apresentação do ambiente *LabVIEW*, entendido como uma ferramenta para desenvolvimento de software, nomeadamente das características mais relevantes da linguagem de programação que lhe está subjacente (gráfica, orientada aos componentes e regulado pelo fluxo de dados).

1. INTRODUÇÃO

O presente trabalho encontra-se inserido no âmbito do projecto INFRACOM / VIRTUAL AUTOMATION da iniciativa intitulada SIPROFIT (Iniciativa para a Dinamização do Desenvolvimento e da Produção de Sistemas Produtivos para a Fileira Têxtil). O objectivo global desta iniciativa consiste na criação de competências tecnológicas na fileira têxtil (Tinturaria, Acabamentos, Confecção, Tecelagem, etc.), concretizando-se na concepção, desenvolvimento e produção de sistemas e equipamentos produtivos.
O objectivo do projecto INFRACOM / VIRTUAL AUTOMATION é o desenvolvimento de uma metodologia e respectivas ferramentas de apoio ao projecto de sistemas de informação orientados ao controlo, para a supervisão de processos nas industrias têxtil e do vestuário, dando suporte à gestão da produção, qualidade e manutenção. Adicionalmente, pretende-se que os sistemas a desenvolver sejam modulares, escaláveis e parametrizáveis. Para a validação do projecto foi desenvolvido um protótipo numa empresa têxtil.

Este artigo foca o desenvolvimento de um conjunto de VIs (instrumentos virtuais) genéricos, no sentido de tornar possível a interligação de um programa de supervisão e de monitorização industrial, construído em *LabVIEW*, com um conjunto de componentes de sistemas de informação, nomeadamente base de dados (SQL), GSM, HTTP, Excel e ferramentas de geração de relatórios (Crystal Reports da Seagate Software). Os componentes são genéricos no sentido de serem um apoio ao *LabVIEW* facilitando a integração da ferramenta num ambiente em que o *sistema de informação de gestão* (***m**anagement **i**nformation **s**ystem* - MIS) também tem de ser considerado.

2. NÍVEIS CIM

Na era da sociedade da informação e do conhecimento, em que a *internet* tornou realidade o comércio (*eCommerce*) e o negócio electrónicos (*eBusiness*), com as organizações a basearem todos os seus processos logísticos na *internet* (numa tentativa de garantir uma satisfação permanente das necessidades dos clientes), as organizações industriais começam a sentir que os processos organizacionais de suporte não têm incluído o seu verdadeiro negócio, que consiste na produção de bens e produtos [1].
É neste contexto de mudança do funcionamento interno das organizações que surge, no seio da própria indústria produtiva, a necessidade de caminhar no sentido de dotar as organizações industriais da capacidade de disponibilizar e controlar informação produtiva, em tempo-real, a todos os níveis e tipos de gestão, em qualquer local e a qualquer momento [2]. A integração das aplicações tempo-real produtivas com o sistema de informação corporativo será, num futuro já próximo, um factor de sobrevivência para qualquer organização industrial que esteja inserida nesta sociedade cada vez mais dependente das tecnologias da informação e das comunicações, possibilitando a migração para o próximo paradigma organizacional, designado de *eManufacturing* (produção/manufactura electrónica) [3].

Integração MIS + ICIS

A monitorização e a supervisão de processos industriais tem exigido o investimento em soluções tecnológicas cada vez mais baseadas em sistemas tempo-real embebidos, especialmente desenvolvidas para interligarem os equipamentos de fabrico aos MISs, para controlo da produção, da qualidade e da manutenção. No entanto, para que a implementação de uma filosofia efectiva de produção integrada por computador (***c**omputer **i**ntegrated **m**anufacturing* - CIM) [4] se torne uma realidade organizacional, mas de uma forma aberta e tecnologicamente descomprometida e não segundo soluções proprietárias e, como tal, hipotecadas em termos da liberdade de evolução ao longo do seu ciclo de vida (alteração de requisitos, manutenção, etc.), é necessário colmatar o tradicional "buraco" existente entre os MISs e os equipamentos finais (*shop-floor*) [5].

O problema da cobertura dos níveis CIM (fig. 1) tem sido tratado no âmbito das comunicações industriais, através do desenvolvimento de protocolos específicos, nomeadamente o *MAP* (***m**anufacturing **a**utomation **p**rotocol*), o *TOP* (***t**echnical and **o**ffice **p**rotocol*) e o *MMS* (***m**anufacturing **m**essage **s**pecification*). No entanto, no âmbito das aplicações, tem existido alguma inoperância quanto a uma cobertura efectiva dos níveis CIM 0 e 1, fundamentais para alimentar os MISs com fluxos de informação verdadeiramente operacionais. Neste âmbito, tem sido possível assistir ao relativo insucesso das aplicações SCADA (***s**upervision*, ***c**ontrol*, ***a**utomation and **d**ata **a**cquisition*), devido, sobretudo, à sua considerável rigidez e inflexibilidade, de que são exemplos o *WINCC*, o *INTOUCH* e o *BridgeVIEW*. É importante esclarecer que os níveis CIM 0 e 1 estão muito bem cobertos no que diz respeito ao controlo físico dos processos, uma vez que os equipamentos industriais têm sido dotados com sistemas embebidos, de forma a tornarem-se cada vez mais automáticos. No entanto, as grandes lacunas consistem nos sistemas para monitorização e supervisão dos processos e equipamentos.

Neste cenário, parece evidente a necessidade de soluções tecnológicas para a interligação facilitada entre os níveis CIM inferiores (0, 1 e 2) e superiores (3 e 4). Estas soluções tecnológicas passam, necessariamente, pela utilização de sistemas tempo-real embebidos, e eventualmente distribuídos, que suportem computacionalmente a implementação de verdadeiros *sistemas de informação industriais orientados ao controlo* (***i**ndustrial **c**ontrol-based **i**nformation **s**ystems* - ICISs), perfeitamente complementares aos MISs dentro das organizações industriais, dado que têm por objectivo principal a gestão do fluxo de infor-

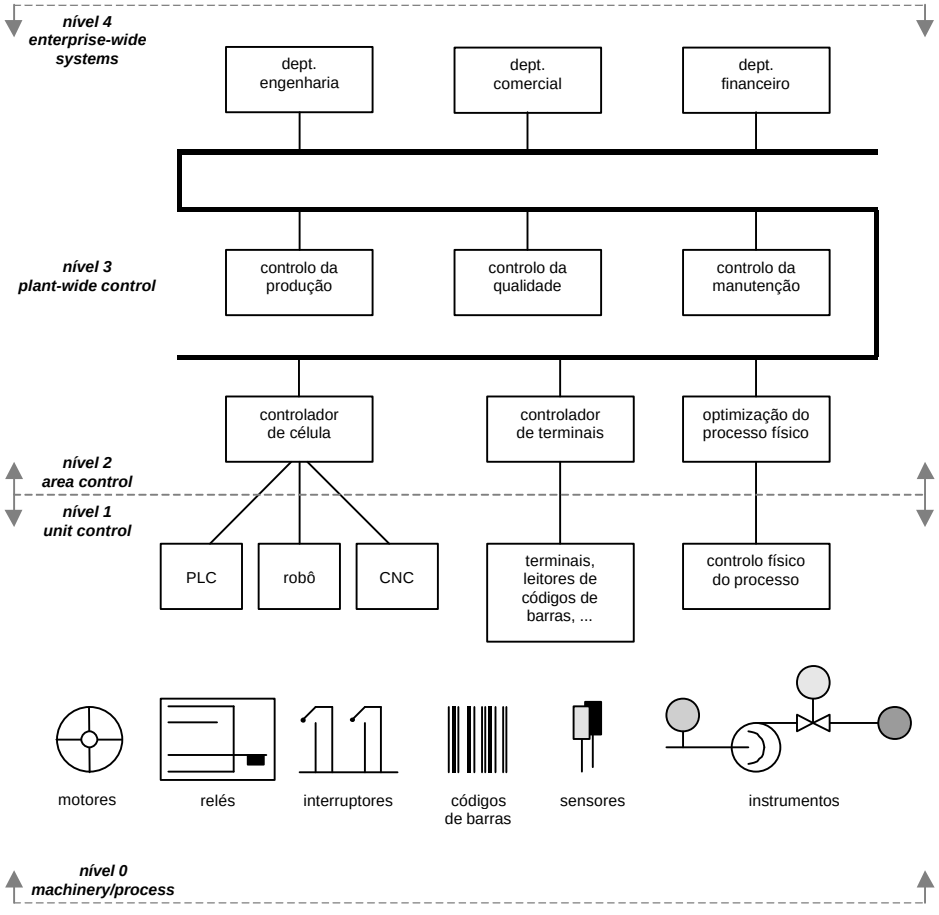

Figura 1 – Níveis CIM.

mação nos níveis CIM inferiores, bem como entre estes e os superiores [6]. Os *sistemas de informação distribuídos não convencionais* consistem, então, no conjunto integrado MIS + ICIS que se pretende que potencie, nas organizações industriais, a eficácia dos princípios defendidos pela corrente designada de *gestão/engenharia dos processos organizacionais* (***b**usiness **p**rocess **m**anagement/**e**ngineering* - BPM/E) [7], uma vez que, assim, as actuais infra-estruturas de tecnologias de informação baseadas em soluções do tipo ERP (***e**nterprise **r**esource **p**lanning*) e POS (***p**lant **o**peration **s**ystem*) conseguem fornecer uma plataforma aplicacional que integra, de uma forma unificada, o controlo e a gestão de toda a informação organizacional [8], essencial para suportar uma verdadeira concepção organizacional orientada pelos processos (*process-oriented business design*), tal como defendido pelo BPM/E.

As estratégias de integração tecnológica MIS + ICIS devem reflectir a consciência de que as divergências entre aqueles dois tipos de sistemas de informação podem ser de quatro tipos distintos [2]:

(1) *temporal*, uma vez que os MISs lidam com escalas de tempo tipicamente na ordem dos dias ou semanas (podendo mesmo chegar aos meses ou anos), enquanto que os ICISs lidam com segundos, ou mesmo com unidades temporais inferiores ao segundo;

(2) *informacional*, uma vez que para os MISs a informação é tratada à luz da semântica transaccional, enquanto que para os ICISs a informação é tratada à luz da semântica de reacção a eventos tempo-real;

(3) *operacional*, uma vez que os MISs suportam directamente operações de planeamento e escalonamento produtivo, enquanto que os ICISs suportam o controlo, a supervisão e a monitorização dos processos e dos equipamentos industriais;

(4) *cultural*, uma vez que o que faz mover os MISs são os negócios organizacionais, enquanto que o "motor" dos ICISs são os processos industriais.

3. O LABVIEW E A ENG. DE SOFTWARE

É consensual afirmar que um ser humano (não deficiente) é, maioritariamente, orientado pela visão, pelo que é frequente desenhar esquemas e diagramas para comunicar ideias e conceitos. Esta realidade parece justificar o facto de as aplicações gráficas serem mais simples de aprender e se apresentarem mais acessíveis a pessoas não técnicas, do que as tradicionais aplicações em que o meio de comunicação é baseado em texto. Desta forma, o *software* gráfico permite àqueles que não dispõem de tempo, formação ou habilidade para programar recorrendo a linguagens textuais convencionais desenvolver os seus próprios programas.

Disponibilizar a não programadores a possibilidade de programar apresenta-se cada vez mais vantajoso e necessário, tendo em conta as exigências de especificidade, as constantes mudanças de requisitos e, também, a carência de programadores em número suficiente para se dedicarem a todas as áreas aplicacionais, por mais simples que sejam as soluções finais [9]. Desta forma, qualquer um pode fazer o seu "pequeno" programa à medida das suas "peculiares" necessidades, sem existir uma grande preocupação com as questões formais de engenharia de *software*. Tal já acontece actualmente, por exemplo, com as folhas de cálculo, dado que qualquer pessoa pode criar as suas tabelas e introduzir as suas fórmulas sem ter que perceber como funciona o "motor" computacional que a interface gráfica esconde. Este não é, de uma forma estrita, o caso dos sistemas de informação industriais, relativamente às actividades de desenvolvimento dos engenheiros. No entanto, a utilização do ambiente *LabVIEW* permite integrar módulos tecnológicos (*hardware*) e módulos algorítmicos (VIs) que elevam o nível de abstracção, de forma a tornar possível o desenvolvimento das soluções finais ao nível do sistema e seguindo uma abordagem CBD (***c**omponent-**b**ased **d**esign*) [10], no suporte directo ao desenvolvimento de componentes (VIs e modelos virtuais de *hardware*) com uma separação formal entre a sua interface e a sua implementação [11].

As vantagens da programação gráfica ultrapassam largamente a simples dispensa de conhecer a sintaxe de uma linguagem de programação textual convencional [12]. Enquanto que com linguagens textuais a programação segue um processo inerentemente linear e que obriga o programador a raciocinar e a exprimir as suas ideias de uma forma algo semelhante como o processador as executa, com linguagens gráficas a conceptualização da aplicação geralmente aproxima-se mais da forma de raciocínio naturalmente seguida pelos humanos (iterativa e irregular). Para ser possível este tipo de abordagem na programação, é necessário que o ambiente de programação gráfica abstraia, o mais possível, os detalhes arquitecturais do processamento computacional, pelo que se torna crucial seguir um meta-modelo de especificação que se adeqúe correctamente ao tipo de programador e à área de aplicação.

LabVIEW

O ambiente *LabVIEW* foi inicialmente desenvolvido pela *NATIONAL INSTRUMENTS* para automatizar aplicações de instrumentação para diversas áreas de engenharia (tipicamente electrónica, electrotécnica e química) e de ciências laboratoriais (essencialmente química e física) [13]. Apesar dos engenheiros e dos cientistas possuírem, normalmente, um raciocínio muito técnico, não se sentem, necessariamente, à vontade em assuntos do âmbito da engenharia informática ou ciências da computação, pelo que esconder dos ambientes de *software* os pormenores típicos da programação textual é uma vantagem enorme para aquele tipo de profissionais. Desta forma, a *NATIONAL INSTRUMENTS* decidiu desenvolver um ambiente de programação gráfica baseada em diagramas de blocos e fluxo de dados que consistem, precisamente, nas ferramentas conceptuais com que os

engenheiros e cientistas estão habituados a lidar no seu dia-a-dia.
Actualmente, o *LabVIEW* constitui uma referência na área, sobretudo porque começou a disponibilizar bibliotecas para uma grande variedade de áreas aplicacionais (processamento de sinal, processamento de imagem, controlo de motores, comunicações industriais, *internet*, acesso a bases de dados, etc.), deixando de ser um ambiente de desenvolvimento exclusivamente dedicado aos sistemas de instrumentação, para se tornar num poderoso ambiente de desenvolvimento de aplicações finais e de prototipagem rápida (porque operacionaliza a execução das especificações) para uma diversidade enorme de profissionais. A versatilidade do seu ambiente de desenvolvimento e o elevado desempenho e a fiabilidade das aplicações finais ficou comprovado, quando a *NASA* recorreu ao *LabVIEW* para desenvolver, com sucesso, uma aplicação capaz de detectar faltas nos circuitos de hidrogénio do *space shuttle COLUMBIA* [9].
Um dos aspectos ao qual é necessário dispensar alguma atenção, consiste na adequação do meta-modelo de especificação seguido pelo *LabVIEW* à área de trabalho específica em que o ambiente está a ser utilizado, (re-)interpretando a forma como as aplicações devem ser desenvolvidas através da:

(1) definição de um sub-conjunto da linguagem a utilizar;
(2) produção de um conjunto de recomendações que constituam um referencial arquitectural para o desenvolvimento de aplicações.

Uma das diferenças significativas que o *LabVIEW* apresenta em relação às linguagens textuais convencionais consiste no facto de seguir uma abordagem orientada aos dados (*data oriented*), no sentido em que a execução dos programas é controlada pelo fluxo dos dados (*data flow*), bem como pela sua disponibilidade (*data driven*). Desta forma, os resultados das computações (*data tokens*) são transportados directamente entre as instruções, sendo os dados produzidos por uma instrução replicados em tantas cópias quantas as instruções que deles necessitam para prosseguir o fluxo computacional. Esta é uma abordagem completamente oposta ao tradicional modelo de computação proposto por *von Neumann* [14], em que a execução dos programas é controlada pelo sequência de instruções escritas pelo programador (*control driven*), sendo, por isso, orientada ao controlo (*control oriented*), no sentido em que a evolução do processamento das instruções depende do fluxo de controlo especificado no programa (*control flow*).
No caso do *LabVIEW*, a abordagem *data flow* reflecte-se, sobretudo, na forma como se especifica:

(1) o paralelismo (concorrência), uma vez que ele é inerente ao fluxo de dados e, frequentemente, independente da replicação estrutural de unidades de controlo;
(2) as reacções aos eventos assíncronos, uma vez que, por vezes, é necessário ter em conta o contexto de computação em que o evento deve surgir;
(3) a hierarquia, uma vez que a violação dos níveis hierárquicos e a terminação de actividades em sub-níveis computacionais se apresentam dificultadas.

Deve, então, reforçar-se a ideia de que em programação *data driven* a semântica dos dados é mais forte do que em programação *control driven*, uma vez que, para além da tradicional semântica de estruturas de informação, existe uma relação directa entre a sua disponibilidade e um controlo implícito sobre fluxo de execução computacional, de uma forma concorrente e assíncrona, em oposição à gestão mecanicista, sequencial e síncrona da programação *control driven*.

4. COMPONENTES E EXEMPLOS

Os componentes *LabVIEW* a apresentar surgiram com o objectivo de facilitar a interligação de soluções desenvolvidas em *LabVIEW*, principalmente orientadas a funções de controlo (ou de forma mais genérica, orientadas para supervisão e monitorização), com conjuntos de recursos de informação, entre os quais se incluem base de dados, HTTP, GSM [15].
Todos estes componentes seguem uma filosofia largamente usada em *LabVIEW*. Primeiro, abre-se uma conexão com um determinado sistema (aplicação ou equipamento), de seguida executa-se o comando ou comandos com esse mesmo sistema e, finalmente, fecha-se a conexão de forma a libertar recursos usados.
Todos os componentes construídos foram colocados numa sub-paleta da paleta *functions*. (fig. 2).

Figura 2 – Paleta Virtual Automation.

Embora tenham sido desenvolvidos diversos conjuntos de componentes, neste artigo e para efeitos de ilustração, apresenta-se em pormenor apenas os componentes *LabVIEW* para interacção com bases de dados (via comandos SQL), com modems GSM e com

arquitecturas de hardware próprias, capazes de obter e enviar informação de e para equipamentos produtivos industriais.

Comandos SQL

Apesar de na ferramenta *LabVIEW* já existirem componentes para lidar com a linguagem SQL, sentiu-se, no âmbito do projecto INFRACOM / VIRTUAL AUTOMATION, a necessidade de melhorar esses componentes, a fim de elevar o nível de abstracção.
Os componentes SQL para interacção com base de dados foram construídos a partir do Toolkit SQL da *National Instruments*. Estes VIs permitem inserir/remover dados de uma tabela de uma base de dados, mas têm como desvantagem o facto de ser necessário ter vastos conhecimentos da linguagem SQL para os usar.
Os componentes construídos (fig. 3 e 4) permitem que, qualquer pessoa que não conheça a linguagem SQL, construa facilmente um programa que manipule dados armazenados numa base de dados relacional.

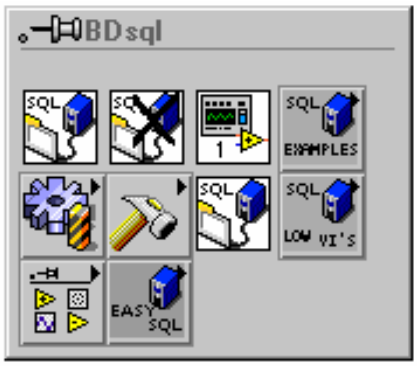

Figura 3 – Paleta dos componentes SQL.

Para ter acesso a uma base de dados (local ou remota), via SQL, é necessário programar um conjunto de parâmetros no ODBC (*Open Database Connectivity*). Esta interface de programação permite que as aplicações tenham acesso a dados em sistemas de gestão de base de dados que utilizem SQL como linguagem de query. Os parâmetros são:

- Nome do DSN (*Data Source Name*). Armazena informação sobre como ligar ao fornecedor de dados;
- Controlador. DLL (*Dynamic Link Library*) que faz o interface entre a aplicação e o sistema de base de dados;
- Nome do utilizador da base de dados (opcional);
- Senha de acesso à base de dados (opcional).

Seguindo a abordagem do *LabVIEW* (abrir conexão, executar comandos, fechar conexão) foi construído um conjunto de VIs SQL, que permite:

- Abrir uma conexão com a base de dados;
- Executar comandos SQL
- Fechar a ligação à base de dados.

Na fig. 5 está representado um VI que permite fazer *queries* à base de dados, sem a necessidade de as escrever textualmente.
O utilizador apenas deverá indicar os campos que quer analisar e as condições de procura a que a *query* estará condicionada.
Os restantes componentes apresentados na fig. 3 têm todos o mesmo propósito (o de fazer *queries* a bases de dados) e foram todos construídos usando a mesma tecnologia e os mesmos princípios, sendo por isso desnecessário apresentá-los pormenorizadamente.

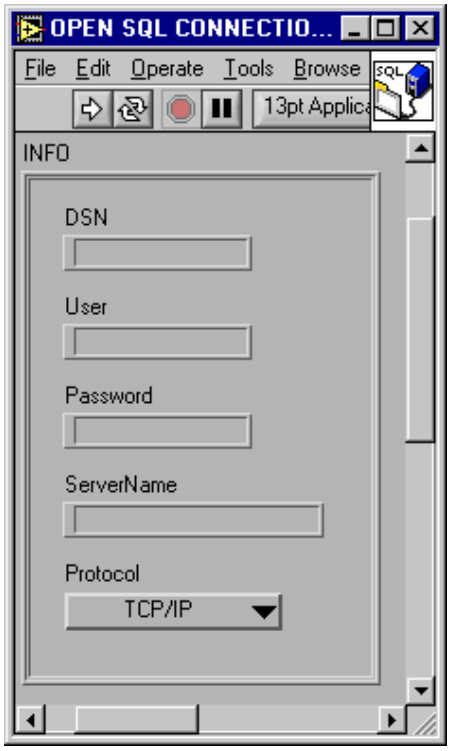

Figura 4 – VI Open SQL Connection.

Mensagens SMS

Neste tópico é apresentado um conjunto de componentes que servem de interface com um modem GSM, de forma a ser possível o acesso a um conjunto de serviços disponíveis numa rede deste tipo.
O objectivo da construção destes componentes é o envio e a recepção de mensagens SMS para telemóveis, endereços de correio electrónico e números de faxes. É pretendida também, a recepção de mensagens SMS para posterior tratamento.
Os componentes (VIs) construídos, e descritos neste tópico, são totalmente compatíveis com todos os modems GSM, que aceitem os comandos standard AT.
A fig. 6 apresenta a paleta onde se encontram todos os VIs construídos para lidar com mensagens SMS e os respectivos modems GSM.
Na fig. 7 apresenta-se o VI *Send AT Command*, que permite o envio de comandos AT via porta série para o modem e a recepção de uma resposta deste. Internamente, tem implementado um mecanismo de *time-out*, que garante que o componente não bloqueia.

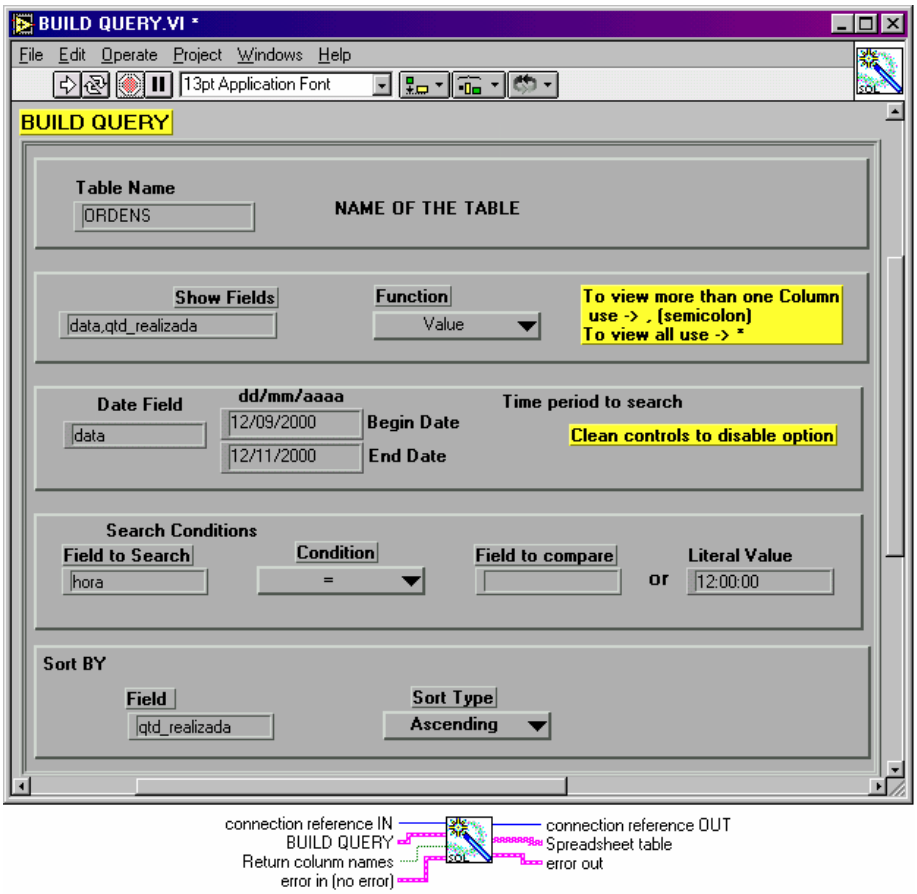

Figura 5 – Componente Easy query.vi.

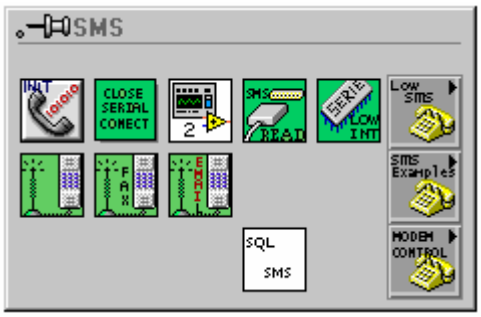

Figura 6 – Paleta dos componentes SMS.

Para interagir com o modem basta especificar a porta série onde se encontra ligado, o tempo disponível até ocorrer um erro de *time-out* e o comando AT a enviar.
O VI retorna o resultado da operação através do indicador *Data Read from DTE*. Se for apenas necessário obter o ACK (*acknowledge*), este está disponível no indicador: *Received ACK*. Se o modem não responder num espaço de tempo inferior ao tempo especificado em *Timeout Limit*, o VI retorna erro de time-out no indicador *Time Out*

Modelo Virtual de FMOTS

A fim de incluir no ambiente *LabVIEW* a possibilidade de serem usados módulos de hardware com software parametrizável (designados FMOTS – ***f****unctional* ***m****odules* ***o****ff-****t****he-****s****helf*), foram desenvolvidos em *LabVIEW* alguns VIs que os representam. Esses VIs são entendidos como modelos virtuais do FMOTS e são usados em *LabVIEW* duma forma bastante abstracta (i.e. sem a necessidade de conhecimentos de hardware). No âmbito do projecto, foi construído um FMOTS e respectivo modelo virtual, para se seja possível monitorizar e controlar um sistema produtivo (neste caso, teares) a partir do *LabVIEW*.
O FMOTS foi desenvolvido a partir do levantamento de requisitos feito numa empresa têxtil e sobre o qual foi especificado o seu comportamento numa linguagem orientada aos objectos (OBLOG). Em paralelo, foi desenvolvido em protocolo de comunicações com base no protocolo CAN (o VAP - *Virtual Automation Protocol)*. Este protocolo descreve a forma de vários FMOTS e CANios (módulos de aquisição e envio de dados de e para os equipamentos produtivos) comunicarem entre si e entre o *LabVIEW* (a partir do componente CANSERVER, desenvolvido no âmbito do projecto Virtual Automation).
Como forma de gerar o código C para o controlador (CANit) foram desenvolvidas em OBLOG regras de geração de código C. Estas regras transcrevem o código desenvolvido em OBLOG (que descreve o comportamento funcional do FMOTS) para código C.
O modelo virtual do FMOTS permite enviar comandos para o FMOTS e receber deste determinada

informação. A fig. 8 apresenta o modelo virtual construído.

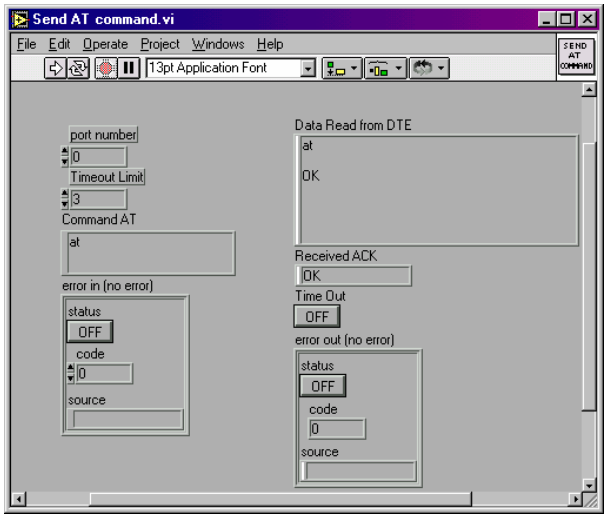

Figura 7 – Vi Send AT command.

De referir que o FMOTS desenvolvido para os teares pode controlar até 25 máquinas.

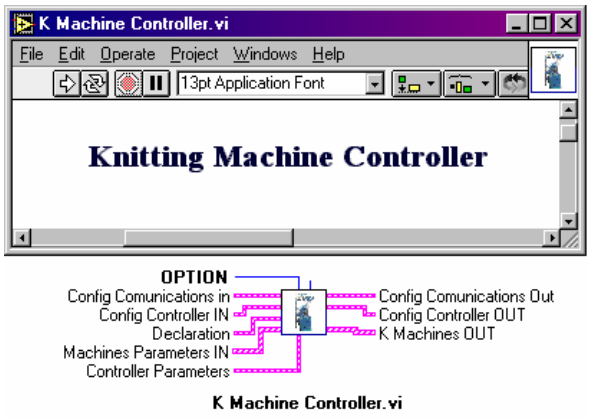

Figura 8 – Modelo Virtual do FMOTS criado.

Para ser possível interagir com o FMOTS é necessário parametrizá-lo.
O protocolo VAP (*Virtual Automation Protocol)* especifica que todos os componentes da rede têm um endereço físico. Neste sentido, o controlo *Declaration* permite especificar os endereços físicos dos CANios e do controlador. O controlo *Controller parameters* contém os parâmetros funcionais do FMOTS (intervalo entre envios de informação para o modelo virtual, tempo de produção do tear). Estes podem ser alterados dinamicamente. O indicador *K Machines OUT* retorna informação sobre a máquina: estado (com erro, parada, a produzir), dados produtivos, resposta a comandos. A fig. 9 mostra alguns dos dados recebidos pelo modelo virtual.

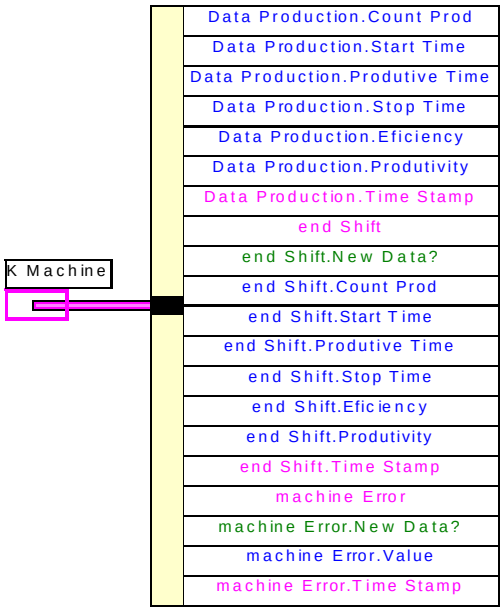

Figura 9 – Dados recebidos pelo modelo virtual do FMOTS.

5. CONCLUSÕES

Como linguagem gráfica, o *LabVIEW* revelou-se uma excelente SLDL, capaz de suportar "perfeitamente" uma abordagem CBD na especificação e desenvolvimento de soluções finais imediatamente executáveis e a um nível de abstracção suficientemente elevado para que um engenheiro de sistemas de informação consiga realizar projectos de sistemas sem que se sinta desconfortável pelo simples facto de estar a lidar com sistemas tempo-real embebidos (uma classe de sistemas tradicionalmente fora do campo de acção daqueles profissionais). Este tipo de utilização do *LabVIEW* exige, no entanto, que sejam definidas algumas regras claras e simples de forma que a programação *data driven* imposta pelo meta-modelo de especificação inerente ao *LabVIEW* não constitua um impedimento semântico à especificação dos necessários paralelismos, reacção a eventos assíncronos, violação de níveis de hierarquia e terminação de actividades, tão frequentes nos sistemas tempo-real embebidos. Como ambiente de desenvolvimento, o *LabVIEW* forneceu um suporte adequado, por exemplo, à construção da *biblioteca virtual automation* e garantiu um tempo de desenvolvimento, para a solução construída, praticamente imbatível, tendo em conta as exigências de comunicação heterogénea (*CAN*, TCP/IP, RS-242 e GSM), de distribuição da solução (vários PCs foram incluídos na solução: PC-VAP, PC de produção, PC de manutenção, PC de planeamento, PC da direcção e servidor *ORACLE*) e da extensão de cobertura (1 *FMOTS*, 3 *CANits*, 24 *CANios* e14 teares) que foram incutidas ao protótipo da solução final.

Como trabalho futuro, prevê-se, para breve, o desenvolvimento de componentes *LabVIEW* que permitam a utilização do protocolo WAP para trocar

informação entre nodos *wireless*, assim como a possibilidade em usar PDAs como pontos de interacção com o sistema de informação que a solução de automação preconiza.

6. BIBLIOGRAFIA

[1] *The New Productivity Factor*, LIPRO Holding AG, 1999.

[2] R. R. Derynck; T. Hutchinson, *Integrating Real-Time Systems with Corporate Information Systems*, The Hewlett-Packard Journal, vol. 50, nº. 1, pp. 26-28, Novembro, 1998.

[3] R. J. Machado, *Metodologias de Desenvolvimento em Projectos de Engenharia de Computadores no Suporte à Implementação de Sistemas de Informação Distribuídos Não Convencionais (Industriais)*, Tese de Doutoramento em Informática / Engenharia de Computadores, Escola de Engenharia, Universidade do Minho, Braga, Novembro, 2000.

[4] J. B. Waldner, *CIM: Principles of Computer-Integrated Manufacturing*, John Wiley & Sons, 1992.

[5] B. Scholz-Reiter,, *CIM Interfaces: Concepts, Standards and Problems of Interfaces in Computer Integrated Manufacturing*, Chapman & Hall, 1992.

[6] R J. Machado, J. M. Fernandes, H. D. Santos, *Sistemas de Informação Industriais Orientados ao Controlo: Perspectivas Metodológicas para Tecnologias Reconfiguráveis*, Ingenium, Revista da Ordem dos Engenheiros, II série, no. 50, pp. 88-92, Jul/Ago, 2000, [ISSN-0870-5968].

[7] A. W. Sheer, *Business Process Engineering: Reference Models for Industrial Enterprises*, Springer-Verlag, 2nd edition, 1994.

[8] A. Oliveira, *A Importância dos Sistemas de Informação para a Indústria*, Sistemas de Informação, APSI, nº. 11, pp. 21-26, 1999.

[9] T. Williams, *Object-Oriented Methods Transform Real-Time Programming*, Computer Design, pp. 101-118, Setembro, 1992

[10] J. Jehander, *Graphical Object-Oriented Programming in LabVIEW*, application note 143, NATIONAL INSTRUMENTS, Outubro, 1999.

[11] R J. Machado, J. M. Fernandes, *A Petri Net Meta-Model to Develop Software Components for Embedded Systems.* 2nd IEEE International Conference on Application of Concurrency to System Design - ACSD'01, Newcastle Upon Tyne, Reino Unido, Junho, 2001, IEEE Computer Society Press. [a publicar].

[12] F. J. Rammig, *Beyond VHDL: Textual Formalisms, Visual Techniques, or Both?*, IEEE European Design Automation Conference (EuroDAC'96), pp. 420-427, Genebra, Suíça, 1996.

[13] R. Bishop, *Learning with LabVIEW*, Addison Wesley, 1999.

[14] K. Hwang, *Advanced Computer Architecture: Parallelism, Scalability, Programmability*, Computer Science Series, McGraw-Hill International Editions, 1993.

[15] M. Carvalho, *Desenvolvimento de componentes genéricos em LabVIEW para integração com sistemas de informação*, Relatório de estágio, DEI, UM, 2000.