III Simpósio Doutoral do Departamento de Informática da Universidade do Minho

"Automatização da Geração de Código para Robôs Industriais Diversos"

Doutorando: Gustavo Vasconcelos Arnold (gva@di.uminho.pt)

Orientador: Pedro Rangel Henriques (DI/UM, prh@di.uminho.pt)
Co-orientador: Jaime Cruz Fonseca (DEI/UM, Jaime.Fonseca@dei.uminho.pt)

Data Início: Novembro de 2002
Data Término: Outubro 2006

Área de Investigação e Desenvolvimento:

Especificação e Processamento de Linguagens

Resumo:

Este relatório apresenta o estado atual da tese de doutoramento, que tem por objetivo desenvolver um ambiente de programação para robôs industriais, que seja amigável e portável. Esta portabilidade diz respeito não apenas a portabilidade do ambiente de programação, mas principalmente a portabilidade do código-fonte. Desta maneira, será possível controlar diversos robôs a partir de um único programa-fonte, facilitando as tarefas de programação e manutenção.

Para atingir este objetivo, um ambiente genérico de programação *off-line* de robôs industriais foi criado: GIRO (Grafcet – Industrial Robots). GIRO tem como foco a modelação do problema a ser resolvido, ao invés da modelação segundo as características do robô. Utiliza o diagrama de especificação de automatismos Grafcet como front-end, mas permite escrever programas diretamente na linguagem do robô.

Através do GIRO: O usuário pode programar robôs em alto ou baixo nível; A portabilidade de código-fonte é garantida; O reuso de código-fonte para robôs diversos é permitido; As tarefas de programação e manutenção são facilitadas; E é fácil de usar! Então GIRO é "*giro*"!

Enquadramento

Today, there are basically two ways for programming industrial robots: (1) by the use of the industrial robot programming language (**IRPL**); (2) or by the use of off-line programming environments (**OLPE**). However, both of them have their own drawbacks:

- although the forerunner languages, such as AML [Tay82] or AL [Muj82], have now been superseded by elaborated robot languages, like ABB Rapid [Abb94], they require detailed description of the motion of every joint in the mechanism in order to execute a desired movement; they require specialized knowledge of the language; the robot programs have limited portability, and significant investment must be made when changing or acquiring a new robot type or simply when upgrading a new controller from the same vendor.

- the most recent off-line programming environments (for instance, Workspace) do not address the issue of sensor-guided robot actions; and they are limited to a robot motion simulator, which provides no advanced reasoning functionality, nor flexibility in the tasks.

According to some principles of software engineering and programming languages evaluating criteria [Sebesta99], it is possible to see that these development environments do not attend the user needs, as can be seen on figure 1.

Principles	**OLPE**	**IRPL**
User-Friendly	YES	NO
Source Code Portability	YES	NO
Expressiveness	NO	YES
Environment Interaction	NO	YES
Specification close to the problem	NO	NO
Reusability	NO	NO

Figure 1: Principles of software engineering and programming languages that must be supported by the programming environments

So, we proposed an integrated, formal and high-level approach to industrial robot programming in [ARN03a]. But this approach evolved, it was modified, and a new one is almost done, named GIRO (Grafcet - Industrial Robots): A Generic Environment for Programming Industrial Robots Off-Line. GIRO attends the above users desires (figure 1), facilitating the programming and maintenance tasks, allowing the reuse of source code.

Our Initial Approach to Industrial Robot Programming

Our initial approach is described diagrammatically in figure 2. It should be supported by the following languages and tools:

- a truly high level and declarative language (Grafcet)
- an easy-to-use front-end
- an intermediate representation (DIR)
- the translators to RS and to DIR
- the code generator
- an automatic generator of the robot code generators
- a robot specification language

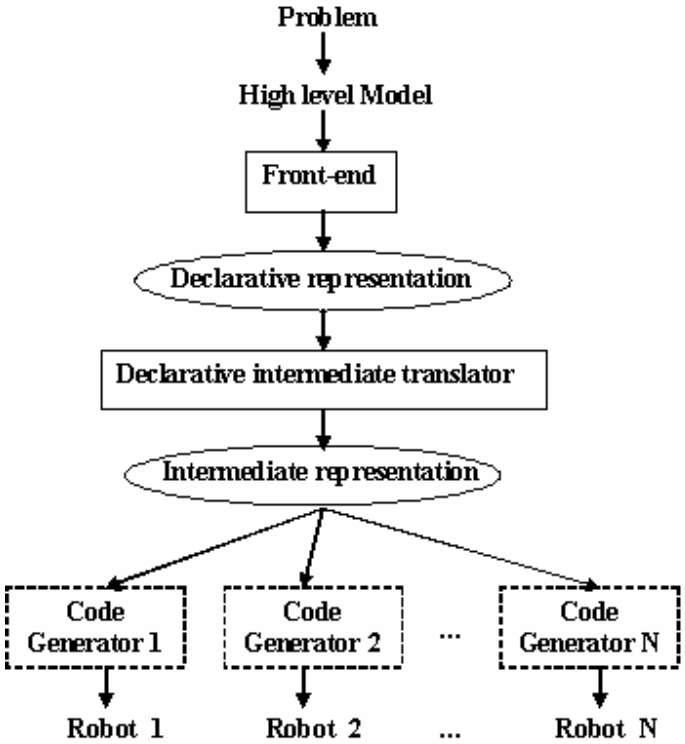

Figure 2. Proposed approach to industrial robots programming.

At the top, there is the problem to be solved. It would be used an adequate modeling technique, responsible for decomposing the problem into simple problems, that would be easily programmed; formal models must be employed to describe data structures and operations necessary to solve the subproblems. To describe formally the overall problem and the subproblems, a truly high level language, close to the specification instead of the robot, should be used. Then we advocate the use of an easy-to-use compiler front-end, like Grafcet, that interprets the specification language and generates a declarative description, that can be translated into an intermediate description for the program specified. An

intermediate representation is used because the front-end must be focused on the specification of the problem, and not on the robot. So, another component, responsible for translating this intermediate code into the code of a robot, should be implemented. Because this compiler back-end is specific for a single robot, it is necessary to have a lot of back-ends, each of them adapted to a specific target (robot's architecture and machine code). To create these code generators, an automatic generator would be used, as can be seen in figure 3. This tool, based on the known intermediate representation, and on the robot specification, that must be included someway, would produce an optimal code generator for the specified robot.

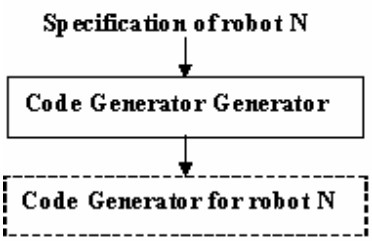

Figure 3. Automatic generator of code generators.

The main objective of this approach was to make the programming task easy, with the possibility of reusing the source code to program different industrial robots, allowing to explore their potentialities.

GIRO's Approach

After some research and development, the initial approach has evolved to GIRO's approach. It happened because one of the main goals was to allow any user to use our environment. But, and at the first time using such a different robot? It should be necessary, for this user, to give the specification of this robot to our code generator generator. And it would not be a simple task. Maybe impossible for this user. So, to facilitate this task, it was decided to change this input. Instead of giving a specification of the robot, it would be easily to give a subset of the robot language grammar. The user can get this subset taking a look at the robot language manual. And with a friendly interface, he could give this grammar in a easy way, just answering some questions.

GIRO's approach can be seen on figure 4. It is supported by the following languages and tools that compose the architecture we want to defend:

- a truly high level and declarative language (Grafcet)
- an easy-to-use front-end
- an intermediate representation (InteRGIRo)
- the translators to InteRGIRo
- the generic compiler
- the robot language grammar subset and symbols editor

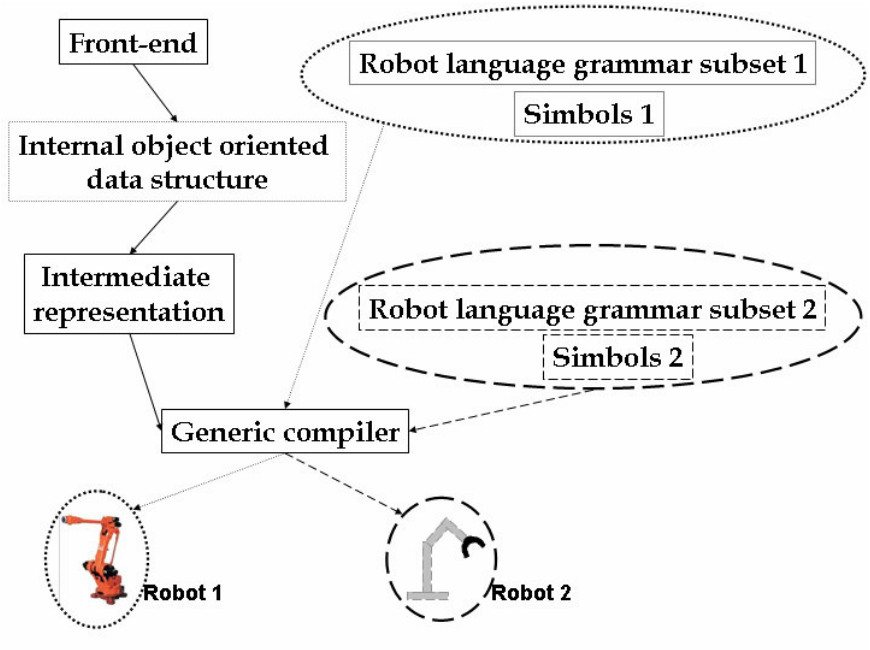

Figure 4. GIRO's approach to industrial robots programming.

As in the initial approach, at the top, there is the problem to be solved. It also would be used an adequate modeling technique, and, to describe formally the overall problem, it will be used an easy-to-use compiler front-end, based on Grafcet, that is a truly high level language, close to the specification instead of the robot, that interprets the specification language and generates an internal data structure, that can be translated into InteRGIRo, an intermediate description for the program specified. Another component, the generic compiler, responsible for translating InteRGIRo into the code of a robot, was implemented. This generic compiler, based on the known intermediate representation, on the subset of the robot language grammar, and on the used symbols (variables, sensors, positions, etc), produces the code for the specified robot. Both the grammar subset and the symbols must be included in a friendly interface, that will be built sooner.

The objective of GIRo extends the objectives of the initial approach, with a easy-to-use interface for describing the robot language grammar subset and symbols. So, any user will be able to control any industrial robot, even if this is a new robot.

Trabalhos Alternativos

- Ambientes de programação off-line, como o Workspace e o X-Arm [bet01]. Podem garantir a portabilidade de código para robôs diversos. Mas para isto é preciso ter um contrato de manutenção com o desenvolvedor deste ambiente. Esta empresa é responsável por construir um tradutor para um novo robô. No entanto, são ambientes de simulação, e não ambientes de programação. Apesar de ser possível marcar pontos por onde o robô deverá se deslocar, e definir os comandos de movimentação, não possue nenhuma flexibilidade, nem permite o acesso aos dados dos sensores. Ou seja, não é expressiva, não permite uma interação com o ambiente, e a programação é focada no robô, e não no problema.
- BERRA, OSCAR, OPEN-R, Mobility e Teambots [ORE00], dentre outras arquiteturas de software existentes para robôs móveis, que podem ser aplicadas em robótica industrial, como a Subsumption Architecture [BRO86]. Entretanto, as implementações existentes são dependentes de hardware, ou seja, foram feitas para um único robô móvel.
- Algumas linguagens de programação de alto nível para robôs móveis, como por exemplo a Functional Robotics (FROB - http://haskell.cs.yale.edu/frob/), e o Potential Use of XML in Robotics, de Sami Raimi. No entanto, novamente foram desenvolvidas para programar um único robô.
- Orocos (http://www.orocos.org/) apresenta uma arquitetura de software para robôs, mas que está a um nível abaixo da proposta desta tese, mais voltada para a criação de primitivas de sistema operacional para controlar robôs.
- Máquinas virtuais, como .NET, JVM, não se aplicam uma vez que o objetivo deste trabalho é programar robôs diversos, e isto inclui robôs antigos, robôs que não são controlados por PCs, ou simplesmente robôs que não possuem recursos de hardware capazes de suportar uma a execução de uma máquina virtual.
- Trabalhos semelhantes para computadores pessoais, como o BURG e o BEG. É preciso ter uma descrição do hardware para o qual será gerado automaticamente um gerador de código. Entretanto, não consegui ter acesso a descrição do hardware do robô, provavealmente por interesses comerciais. Além disto, um dos objetivos é tornar fácil a utilização desta ferramenta para os usuários, mas descrever a arquitetura de alguma máquina não é uma tarefa simples. Para isto seria preciso uma pessoa da área de compiladores, e não um usuário comum.
- IRL (Industrial Robot Language) e outras tentativas de criar uma linguagem padrão para robos industriais não vingaram. Existem, mas não são utilizadas. Isto ocorre provavealmente: ou por não estar claro, ainda, os tipos de comandos que devem ser considerados como básicos, ou padrões; ou devido aos fabricantes de robôs não as adotarem.
- RobotScript. Uma linguagem universal para a programação de qualquer robô industrial. No entanto, é necessário adquirir o Universal Robot Controller (URC) para utilizar o RobotScript e controlar o robô. Este URC substitui o controlador do robô. Para funcionar, alguém terá que desenvolver os drivers necessários para que a URC controle determinado robô. Como sua a arquitetura é aberta, qualquer empresa pode desenvolver estes drivers. Esta solução transfere o problema da portabilidade de código, que era da linguagem de programação, para os drivers de comunicação com o robô.

Desenvolvimento

Antes de começar...

Algumas atividades foram desenvolvidas na Universidade do Minho antes do início do doutoramento, como as que se seguem:

1. Discussão junto ao orientador (Pedro Rangel Henriques) e co-orientador (Jaime Cruz Fonseca) sobre os assuntos a serem estudados para um correto andamento desta pesquisa, uma vez que este trabalho abrange conteúdos de áreas afins, mas diferentes (Informática e Eletrônica Industrial).
2. Limitação da área de atuação do ambiente a ser desenvolvido: Será definida uma família de robôs industriais para a aplicação do tema. A família de robôs industriais a ser usada será aquela a ser disponibilizada ao laboratório de Eletrônica Industrial do Campus de Gualtar.
3. Identificação da aprovação do tema junto as comunidades acadêmicas envolvidas na temática deste trabalho, uma vez que foram aceites e apresentados 2 artigos: um no **Encontro Científico do Festival Nacional de Robótica (Robótica 2002)**, em Aveiro, Portugal [ARN02a]; e outro no **Simpósio Brasileiro de Linguagens de Programação 2002 (SBLP 2002)**, no Rio de Janeiro, Brasil [ARN02b].
4. Estudo e verificação da possibilidade de utilizar a Linguagem RS como linguagem para controlar robôs industriais. Deste estudo foram feitos mais dois artigos, além dos apresentados no item anterior: um submetido ao **Crossroads ACM Student Magazine** que foi rejeitado por ser complexo para o público da revista [ARN02c]; e outro para o **VII Simpósio de Informática**, que realizou-se em Uruguaiana, Brasil [ARN02d].
5. Identificação da necessidade do aluno assistir a algumas aulas no curso de Mestrado do Departamento de Eletrônica Industrial, para que o mesmo se aprofunde e se familiarize com alguns termos de Robótica.
6. Participação em eventos: como o **Robótica 2002**, em Aveiro; o **Simpósio Brasileiro de Linguagens de Programação**, no Brasil; e o **International Interdisciplinary Seminar on New Robotics, Evolution and Embodied Cognition (IISREEC)**, em Lisboa. Neste último o aluno discutiu assuntos relacionados ao seu doutoramento com pesquisadores de renome na área de robótica, como Rodney Brooks e Rolf Pfeifer.
7. Participação em reuniões do grupo (GEPL).
8. Aprendizagem e utilização da ferramenta Workspace, na qual é possível implementar programas que serão simulados nesta ferramenta. Esta ferramenta simula vários dos principais robôs industriais existentes.
9. Pesquisa bibliográfica no intuito de identificar o estado da arte em linguagens de programação para robôs industriais. Esta etapa ainda está sendo realizada.

Primeiro ano:

As atividades desenvolvidas na Universidade do Minho no âmbito deste projeto de doutoramento encontram-se descritas na seguinte lista:

1. Confrontação do tema junto as comunidades acadêmicas envolvidas na temática deste trabalho, uma vez que foram aceites e apresentados mais 2 artigos: um no **IEEE 11th International Conference on Advanced Robotics (ICAR 2003)**, em Coimbra, Portugal [ARN03a]; e outro no **Fourth Congress of Logic Applied to Technology (LAPTEC 2003)**, em Marília - São Paulo, Brasil [ARN03b].
2. O aluno assistiu algumas aulas no curso de Mestrado do Departamento de Eletrônica Industrial, para que o mesmo aprofundasse e se familiarizasse com alguns termos de Robótica. As aulas assistidas foram: Sensores e Actuadores; e Tecnologia Robótica.
3. Participação em reuniões do grupo (GEPL).
4. Orientação do trabalho de Especialização **Análise da Utilização de Equipes de Robôs em Operações de Busca**, do Curso de Pós-Graduação em Sistemas Distribuídos da Universidade Católica do Salvador, no qual fiz parte da banca avaliadora em fevereiro de 2003, em Salvador, Bahia - Brasil.
5. Elaboração de uma abordagem de desenvolvimento de software para robôs industriais, que abrange todos os estágios da programação, desde a modelagem até a geração de código para o robô industrial. É composta por um front-end (Grafcet), uma linguagem declarativa (RS), uma representação intermediária (DIR) e por um gerador automático de geradores de código, que é o principal objetivo desta tese de doutoramento.
6. Desenvolvimento de exemplos a fim de validar a abordagem proposta. Estes exemplos identificaram as similaridades entre os componentes da abordagem proposta, visando garantir que os mesmos sejam interligados.
7. Início da escrita da tese de doutoramento.
8. Pesquisa bibliográfica referentes aos seguintes assuntos:
   - Fundamentos de robótica industrial, através de livros de robótica, como [GRO87] e [REH97].
   - Linguagens de programação para robôs industriais, através de alguns manuais de programação de alguns robôs industriais, como em [Abb94] e [NAC94].
   - Arquiteturas de software para robôs industriais, bem como os componentes importantes nestas arquiteturas que devem ser tratados/incluídos na abordagem proposta, como o Workspace (www.workspace5.com), NIST Component Specifications [Mess99], BERRA, OSCAR, OPEN-R, Mobility e Teambots [ORE00], dentre outras arquiteturas de software existentes para robôs móveis, que podem ser aplicadas em robótica industrial, como a Subsumption Architecture [BRO86].
   - Linguagem diagramática para especificação de automatismos sequenciais, GRAFCET [Grafcet], bastante utilizada entre as pessoas da área de robótica industrial, e na qual será baseada a interface gráfica proposta na abordagem.
   - MIR, Machine Independent Representation, uma representação intermediária desenvolvida por Paulo Matos [Pmatos], que visa garantir tanto uma independência do programa fonte com relação aos robôs que o irão executar, como garantir uma independência destes robôs frente as linguagens de programação a serem criadas para programá-lo. Esta representação está em

constante evolução, e neste momento se chama DIR – Dolphin Intermediate Representation.

- Outros, que apesar de envolverem linguagens de programação e robótica, e fazerem parte do estado da arte neste assunto, não se enquadram na abordagem proposta, como por exemplo a **Functional Robotics** (FROB - http://haskell.cs.yale.edu/frob/), e o **Potential Use of XML in Robotics**, de Sami Raimi. Estes não se enquadram pois apenas foram desenvolvidas linguagens de alto nível para programar um único robô, diferente da minha proposta, que é ter uma única linguagem para programar robôs diversos. Outros, como o [Orocos] (http://www.orocos.org/), apresenta uma arquitetura de software para robôs, mas que está a um nível abaixo da minha proposta, mais voltada para a criação de primitivas de sistema operacional para controlar robôs.

Segundo ano:

Durante este segundo ano, o bolseiro se dedicou mais ao estudo de ferramentas para a implementação do ambiente proposto em sua tese de doutoramento, em sua implementação, validação e testes, e na escrita de sua dissertação.

As atividades desenvolvidas na Universidade do Minho no âmbito deste projeto de doutoramento, e segundo o calendário proposto, encontram-se descritas na seguinte lista:

1. Discussões junto ao orientador (Pedro Rangel Henriques) e co-orientador (Jaime Cruz Fonseca) para um correto andamento desta pesquisa, uma vez que este trabalho abrange conteúdos de áreas afins, mas diferentes (Informática e Eletrônica Industrial).
2. Assim como em 2003, o aluno participou no Simpósio Doutoral do Departamento de Informática da Universidade do Minho de 2004 (realizado no período de 5 a 7 de janeiro de 2005), onde o aluno apresentou o tema bem como o estado atual do seu trabalho de doutoramento.
3. Participação em reuniões do grupo (GEPL), em palestras/seminários do Departamento, bem como nas Jornadas de Informática da Universidade do Minho (JOIN 2004).
4. Continuação da leitura dos artigos atuais da área (revisão bibliográfica) para suporte à escrita da dissertação de doutoramento.
5. Escolha da linguagem Java para a implementação do ambiente de programação proposto (cf. relatório do ano anterior) e dos tradutores. Anteriormente, se pensou em utilizar o Visual C, mas como o objetivo a longo prazo é o de disponibilizar esta interface na internet, de forma a tornar possível a geração de geradores de código independente de plataforma, a escolha foi para o Java.
6. Estudo sobre interfaces gráficas na linguagem Java. Mais especificamente sobre o Java SWING, utilizado no desenvolvimento da interface gráfica do ambiente proposto.
7. Modelação segundo UML do Grafcet e da linguagem RS.
8. Decisão sobre a constituição do front-end, o qual será composta por três camadas: a primeira responsável pelas propriedades e atributos de cada elemento do Grafcet e da Linguagem RS; a segunda responsável pelas propriedades gráficas de cada um

destes elementos; e a terceira responsável pela interligação destes elementos, pela interface com o usuário, bem como por disparar os tradutores entre as diversas linguagens envolvidas.

9. Implementação em Java das classes criadas em UML.
10. Desenvolvimento do tradutor:
    - GRAFCET - RS
11. Arranque do desenvolvimento do tradutor:
    - RS - DIR
12. Estudo da linguagem de programação do robô industrial RV-M2, da Mitsubishi. Em seguida será desenvolvido um tradutor MIR – Robô RV-M2, a fim de finalizar o protótipo do ambiente de programação, e partir para o desenvolvimento do gerador automático de geradores de código.
13. Escrita de alguns capítulos da dissertação de doutoramento.

Terceiro ano:

Durante este terceiro ano, o bolseiro se dedicou mais a implementação de tradutores, definição de uma linguagem intermediária, e ao estudo de alguns robôs industriais.

As atividades desenvolvidas na Universidade do Minho no âmbito deste projeto de doutoramento, e segundo o calendário proposto, encontram-se descritas na seguinte lista:

1. Discussões junto ao orientador (Pedro Rangel Henriques) e co-orientador (Jaime Cruz Fonseca) para um correto andamento desta pesquisa, uma vez que este trabalho abrange conteúdos de áreas afins, mas diferentes (Informática e Eletrônica Industrial).
2. Confrontação dos resultados junto as comunidades acadêmicas envolvidas na temática deste trabalho, através de apresentação de um artigo intitulado "**A Graphical Interface Based on Grafcet for Programming Industrial Robots Off-Line**", em Barcelona, no âmbito do **2nd International Conference on Informatics in Control, Automation and Robotics (ICINCO)** (setembro, 2005).
3. Participação em reuniões do grupo (GEPL), em palestras/seminários do Departamento, bem como nas Jornadas de Informática da Universidade do Minho (JOIN 2005).
4. Continuação da leitura dos artigos atuais da área (revisão bibliográfica) para suporte à escrita da dissertação de doutoramento.
5. Construção do ambiente integrado GIRo (Grafcet – Industrial Robots), composto por uma ferramenta gráfica baseada no Grafcet (PaintGraf)), uma representação intermédia (InteRGIRo), e dos tradutores: entre a especificação Grafcet e a linguagem RS; entre a especificação Grafcet e a InteRGIRo; e outro, genérico, entre a InteRGIRo e um qualquer robô industrial.
6. Continuação do desenvolvimento do projeto subjacente a tese: concepção, desenho, implementação e testes da plataforma de compilação.
    a) Configuração e correção do compilador RS, para que o mesmo pudesse ser chamado e executado a partir do ambiente gráfico para edição/compilação de programas em Grafcet (ambiente este apresentado no relatório anterior). Ao contrário do previsto, o tradutor da linguagem RS para a representação

intermédia não foi desenvolvido, uma vez que o autômato gerado pela linguagem RS corresponde a estrutura de dados interna do programa Grafcet desenvolvido neste ambiente.

b) Definição de uma representação intermédia. Como os tradutores atuam entre a interface gráfica e as linguagens de programação de robôs industriais, foi definida uma representação capaz de descrever os comandos basicos das linguagens destes robôs, de maneira a permitir que qualquer programa desenvolvido na interface gráfica seja traduzido para qualquer linguagem de programação de robôs industriais. Esta representação foi definida com base nas linguagens estudadas neste doutoramento, e em linguagens já conhecidas pelo doutorando.
c) Desenvolvimento de um tradutor entre a linguagem gráfica (Grafcet) e a representação intermédia.
d) Definição, e implementação, de um tradutor genérico entre a representação intermédia e qualquer robô industrial. Para isto, é preciso descrever alguns dos comandos da linguagem de programação do robô industrial em questão segundo a gramática definida no ambiente GIRo, juntamente com símbolos que podem corresponder a pontos no qual o robô se deslocará, sensores, ou qualquer outra entrada/saída digital.
e) Desenvolvimento de programas de teste, para validar o editor gráfico de Grafcet, a representação intermédia, e a tradução.
f) Estudo da linguagem VAL, para manipulação do robô industrial Puma (Staubling), a fim de identificar os comandos básicos e a estrutura dos programas necessários para controlar este robô.
g) Estudo da linguagem de programação do robô Mitsubishi RV-M2, a fim de identificar os comandos básicos e a estrutura dos programas necessários para controlar este robô.
h) Descrição dos comandos das seguintes linguagens de programação para robôs industriais, segundo a gramática do ambiente GIRo:
   - Puma (Val).
   - Puma (Val 2).
   - Mitsubishi RV-M2.
   - Sony.
   - ABB.
i) Testes e validação no robô Puma.

Quarto ano:

Em termos gerais, os objetivos para este quarto ano são:

1. Continuar a pesquisa bibliográfica para constante atualização a respeito do estado da arte neste tema.
2. Continuar os estudos em robótica, seja participando de eventos relacionados com a robótica, seja através de leitura.
3. Continuar os estudos com relação a geração automática de geradores de código.

4. Concluir o desenvolvimento do projeto, incluindo a implementação e testes da plataforma.
5. Continuar, e finalizar, a escrita da tese.

Detalham-se, abaixo, as tarefas técnicas, específicas, que serão concretizadas para cumprir os objetivos acima.

1. Realizar testes nos robôs industriais.
2. Desenvolver um ambiente amigável, de forma a simplificar ao máximo a definição da gramática necessária para mapear a representação intermédia na linguagem do robô industrial. O ideal seria tornar qualquer usuário de robô industrial capaz de inferir esta gramática, através deste ambiente amigável.
3. A fim de continuar a validação do ambiente proposto e implementado, será dada continuidade ao desenvolvimento de gramáticas, com específicos testes, para outros robôs industriais que venha a ter acesso, seja pessoalmente, por bibliografia, ou pela internet.
4. Se possível, serão definidas gramáticas para robôs móveis, a fim de aumentar a abrangência de uso do ambiente proposto, numa área diferente.
5. Finalizar a escrita da tese.
6. Apresentar e defender esta tese.

Além destes objetivos, um outro objetivo que, apesar de não ter nada haver com esta tese, poderá acarretar algumas "turbulências" neste quarto ano: participar do nascimento e dos cuidados de Gabriela. Esta previsto nascer no início de março, mas como não queria correr riscos de perder o nascimento, me ausentei deste III Simpósio Doutoral e estou, que chato, em João Pessoa, no nordeste do Brasil...

Recursos

Os únicos recursos necessários, além de um computador e uma secretária, foram alguns robôs industriais, a fim de testar e validar a implementação feita. Quatro robôs foram disponibilizados pelo DEI, um Puma, um Mitsubichi, um Sony e um ABB.

Dificuldades

Uma das maiores dificuldades que encontrei foi justamente na confrontação dos resultados junto as comunidades acadêmicas envolvidas na temática deste trabalho. Quando participei de eventos na área de linguagens de programação (por exemplo, o SBLP), gostavam do trabalho mas não era gerada uma maior discussão, pois os participantes pouco, ou nada, conheciam a respeito de robótica industrial. E quando participei de eventos na área de robótica industrial (por exemplo, o IEEE ICAR), também gostavam, mas acontecia o mesmo: os participantes pouco, ou anda, conheciam a respeito de linguagens de programação.

Para piorar esta situação, eu não encontrava nenhuma referência sobre minha área de doutoramento. Encontrei algumas referências muito antigas (dos anos 70), que buscavam desenvolver um ambiente de programação amigável, mas nunca encontrei os resultados finais, ou seja, não soube se conseguiram ou não desenvolver, os problemas, entre outras informações.

Em 2005 fui a uma conferência "menor" que a IEEE ICAR mas muito melhor para mim. Obtive bons resultados na avaliação do artigo, e boas discussões na apresentação do artigo. O motivo de não ter encontrado nenhuma referência nesta área é justamente porque no fim dos anos 70 ou início dos anos 80 pararam as pesquisas na área de linguagens de programação para robôs industriais. A justificativa era que não valia a pena investir em pesquisa nesta área uma vez que um programa é feito uma vez e executado durante anos por um robô. Porém, este não aprimoramento nesta área acarreta justamente na dificuldade de programação e na dependência do fabricante. Quem sabe se isto não causou um um não desenvolvimento adequado dos robôs industriais?

Avaliação

Neste momento, a comparação com outras propostas não pode ser feita exatamente por não ter encontrado outras soluções, com as mesmas características, para o problema a ser tratado neste doutoramento. Mesmo para robôs móveis, as soluções podem ser amigáveis, mas são dedicadas para um robô. Uma solução amigável, que garanta a portabilidade de código-fonte para robôs ainda não foi encontrada.

Trabalhos semelhantes para computadores pessoais existem, como o BURG e o BEG. Entretanto, não consegui ter acesso a descrição do hardware do robô, e assim não poderia utilizar uma solução semelhante a estas. E como o objetivo que se pretende é tornar fácil a utilização desta ferramenta para os usuários, não fazia sentido o usuário ter que, para utilizar o mesmo código-fonte, fornecer a descrição do novo hardware segundo uma gramática pré-definida. Para isto seria preciso uma pessoa da área de compiladores, e não um usuário comum.

A utilização de máquinas virtuais não seria adequada, pois o objetivo é gerar código para robôs diversos, e alguns destes robôs são antigos, não são controlados por um PC, e não possuem um hardware capaz de suportar a utilização de uma máquina virtual.

Desta maneira, este trabalho é válido, é inédito e não encontrei soluções alternativas que pudesse me basear para o seu desenvolvimento. E para comprovar a importância deste trabalho em robótica industrial, os três artigos submetidos a conferências de robótica foram aceitos. Mesmo que somente o último, submetido a ICINCO'2005, tenha fornecido um excelente feedback (visto ser uma conferência de informática e robótica, com participantes que conhecem tanto de robótica como de linguagens de programação), as outras acharam o trabalho interessante e necessário.

Uma grande valia deste trabalho é permitir que o usuário crie, e utilize, os comandos do robô que ele quiser. Algumas tentativas de criar padrões de linguagens de programação para robôs não vingaram exatamente por não estar claro, ainda, os tipos de comandos que devem ser considerados como básicos, ou padrões.

Contribuições

As principais conclusões deste trabalho são:

- Abrange todas as etapas da programação, desde a modelação do sistema até a geração de código para o robô;
- Uma interface gráfica, baseada no Grafcet, é utilizada como front-end;
- Foi desenvolvida uma representação intermédia, chamada InteRGIRo, para facitiar o processo de compilação;
- Alguns tradutores foram desenvolvidos. Um para traduzir uma representação em Grafcet para a InteRGIRo, e outro, o Compilador Genérico, que traduz a representação InteRGIRo em qualquer linguagem de programação que venha a ser utilizada por um robô industrial;
- Os tradutores e a interface gráfica foram desenvolvidos em Java pois, como um dos objetivos é garantir portabilidade de código fonte, é também importante que o ambiente seja independente de plataforma;
- A programação de robôs industriais torna-se mais simples, sendo possível programar qualquer robô a partir de uma representação em Grafcet;
- Alguns estudos de caso já foram feitos e validados até a data de 8 de fevereiro, mais precisamente gerando código para as linguagens VAL1 e VAL2 (robô Puma) e para a linguagem do robô Mitsubichi Movemaster RV-M2. Outros estudos de caso serão feitos e validados até o dia 16 de fevereiro, mais precisamente gerando código para os robôs ABB e Sony;
- Satisfaz todos os princípios apresentados na figura 1;
- E para o GIRo ficar completamente amigável, serão desenvolvidos front-ends para descrição das estruturas de programa e de controle da linguagem do robô destino, bem como para a descrição das instruções e símbolos. Neste momento estas descrições encontram-se em ficheiros texto.

Os principais motivos que fizeram com que as linguagens de programação de robôs industriais não evoluíssem como as linguagens de programação dos computadores são: interesses econômicos; e a dificuldade de definir um conjunto padrão de comandos utilizados pelos robôs. Algumas tentativas de criação de linguagens padrão não vingaram provavealmente por estes motivos. E este trabalho não terá estes problemas porque: não existe depêndencia de nenhuma empresa, pois as únicas informações necessárias são encontradas no manual da linguagem do robô; e o usuário pode utilizar os comandos que ele achar necessário (se não tiver sido definido previamente, ele próprio é capaz de criar), e desta maneira não são definidas instruções padrão.

Publicações

1. ARN05, Gustavo V. Arnold and Pedro R. Henriques and Jaime C. Fonseca, A Graphical Interface based on Grafcet for Programming Industrial Robots Off-Line, Proceedings of the 2nd International Conference on Informatics in Control, Automation and Robotics, 2005, v 3, p 113-118, Barcelona, Espanha.

   Abstract:
   This paper presents the current development stage of our approach to industrial robot programming, that is the graphical interface for our environment, that is based on the well-known Grafcet. Our approach focus on the modelling of the system, rather than on the robot. So, it will improve the programming and maintenance tasks, allowing the reuse of source code.

2. ARN03a, Gustavo V. Arnold and Pedro R. Henriques and Jaime C. Fonseca, A Development Approach to Industrial Robots Programming, Proceedings of the 11th International Conference on Advanced Robotics, 2003, v 1, p 167-172, Coimbra, Portugal, Institute for Systems and Robotics, University of Coimbra.

   Abstract:
   This paper proposes a development approach to industrial robot programming, that includes: a truly high level and declarative language; an easy-to-use front-end; an intermediate representation; an automatic generator of the robot code generators. So, we introduce a new paradigm to program industrial robots, that focus on the modeling of the system, rather than on the robot. It will improve the programming and maintenance tasks, allowing the reuse of source code, because this source code will be machine independent.

3. ARN02b, G. V. Arnold and P. R. Henriques and J. C. Fonseca, Uma Proposta de Linguagem de Programação para Robótica, Anais do VI Simpósio Brasileiro de Linguagens de Programação (SBLP'02), 2002, p 134-143.

   Abstract:
   This paper presents some existing problems in robotics programming languages and a possible solution using a reactive and synchronous language called RS. An implementation will show how the RS language can facilitate the robot's programming, in this case, industrial robots. However, it should be necessary to create an intermediate module responsible for translating a declarative language to assembly code, in different robots, granting code portability.

4. ARN02a, G. V. Arnold and G. R. Librelotto and P. R. Henriques and J. C. Fonseca, O Uso da Linguagem RS em Robótica, Electrónica e Telecomunicações, 2002, p 501-508, São Paulo, SP, Brasil, Revista do Departamento de Electrónica e Telecomunicações da Universidade de Aveiro.

Abstract:
This paper presents a Reactive and Synchronous language, called RS, as a tool to develop programs for robots, industrial or mobile, creating a higher abstraction level during the programming task. The RS language is simple to use and has constructions to express parallelism and distribution; those are important features that facilitate the development and maintenance of this kind of programs.

5. ARN03b, Gustavo V. Arnold and Giovani R. Librelotto and Pedro R. Henriques, RS Compiler, Proceedings of the Fourth Congress on Logic Applied to Technology (LAPTEC'2003), 2003, Marília, SP, Brasil.

Abstract:
The RS language is intended for the programming of reactive kernels, which are the central and most difficult part of a reactive system. It is simple to use and has constructions to express parallelism and distribution. This paper presents the most recent version of RS language and RS compiler, showing its main features, the improved resources, which make this language a useful tool to develop reactive systems, like programs that control robots, creating a higher abstraction level during the programming task.

6. ARN02d, G. V. Arnold and G. R. Librelotto and P. R. Henriques, Linguagem RS: Um Estudo de Caso, Anais do VII Simpósio de Informática e II Mostra Regional de Software Acadêmico, 2002, PUCRS - Câmpus de Uruguaiana, RS – Brasil.

Abstract:
Este artigo destina-se a apresentar a linguagem Reativa Síncrona RS como ferramenta para o desenvolvimento de software para sistemas controladores e veículos móveis, criando um mais alto nível de abstração no desenvolvimento de programas. Como estudo de caso, mostra-se a especificação de uma auto-estrada inteligente em RS, que contém tanto um sistema controlador (da estrada) como veículos autônomos móveis. A linguagem RS tem características que facilitam bastante o desenvolvimento e manutenção de programas deste tipo, visto que é uma linguagem reativa, simples, paralela e distribuída, e de tempo real.

Bibliografia

Abb94, Rapid Reference Manual 3.0, ABB Flexible Automation AB, 1994.

ARN98a, G. V. Arnold and S. S. Toscani, Using the {RS} Language to Control Autonomous Vehicles, Workshop on Intelligent Components for Vehicles, 1998, p 465-470, Seville, Spain, International Federation of Automatic Control.

ARN98b, G. V. Arnold, Uma Extensão da Linguagem RS para o Desenvolvimento de Sistemas Reativos Baseados na Arquitetura de Subsunção, CPGCC, UFRGS, 1998, Dissertação de Mestrado, Porto Alegre, Brasil.

ARN00, G. V. Arnold, Controle de uma Célula de Manufatura Através da Linguagem RS Estendida, Anais do I Congresso de Lógica Aplicada à Tecnologia (LAPTEC'2000), 2000, p 145-163, São Paulo, SP, Brasil.

ARN02c, G. V. Arnold and G. R. Librelotto and P. R. Henriques, The RS Language Tutorial, Universidade do Minho, 2002, Relatório Técnico.

ARN02e, G. V. Arnold and P. J. Matos and P. R. Henriques, A Simple and Efficient Solution for the Register Allocation Problem, Universidade do Minho, 2002, Relatório Técnico.

Be01, R. W. de Bettio and F. L. S. Garcia, Movimentação de Braços Robóticos em VRML utilizando uma linguagem de programação X-Arm. In: INFO 2001 - 4TH SBC SYMPOSIUM ON VIRTUAL REALITY, 2001, Florianopolis. **SVR.** 2001.

BRO86, R. A. Brooks, A Robust Layered Control System for a Mobile Robot, IEEE Journal of robotics and Automation, 1986,v RA-2, nº 1, p 14-23.

ES89, H. Emmelmann and F. W. Schroer, BEG - a generator for efficient back ends, Proceedings on Programming Language Design and Implementation, 1989, v 24, Oregon.

FH95, C. Fraser and D. Hanson, A Retargetable C Compiler: design and implementation, Addison Wesley Publishing Company, 1995.

FHP91, C. Fraser and R. Henry and T. Proebsting, BURG - fast optimal instruction selection and tree parsing, SIGPLAN Notices, 1991, v 27, p 68 – 76.

FRA77, C. W. Fraser, Automatic Generation of Code Generators, Yale University, 1977, Phd Dissertation, New Haven.

FW88, C. W. Fraser and A. L. Wendt, Automatic Generation of Fast Optimizing Code Generators, Proceedings of the ACM SIGPLAN'88 Conference on Programming Language Design and Implementation, 1988, 79 -- 84, Atlanta, Georgia.

Grafcet, O Grafcet -- Diagrama Funcional para Automatismos Sequenciais, Telemec, Portugal, 1982.

Grafcet2000. Grafcet Homepage, http://www.lurpa.ens_cachan.fr/grafcet.html.

GRO87, M. P. Groover, Automation, Production, Systems, and Computer-Integrated Manufacturing, Prentice Hall, Inc., 1987.

HOP79, J. E. Hopcroft and J. D. Hullman, Introduction to Automata Theory, Languages and Computation, Addison-Wesley, 1979, Massachussetts, USA.

JML91, R. E. Johnson and C. McConnell and J. M. Lake, The RTL System: A framework for code optimization, Proceedings of the International Workshop on Code Generation, 1991, p 255 - 274, Dagstuhl, Germany, May.

MUC97, S. Muchnick, Advanced Compiler Design and Implementation, Morgan Kaufmann Publishers, 1997.

Muj82, M. S. Mujtaba and R. Goldman and T. Binford, Stanford's AL Robot Programming Language, Computers in Mechanical Engineering, 1982.

NAC94, Ar Controller Operating Manual, Nachi Fujikoshi Corporation, Nachi-Fujikoshi Corporation, 1st edition, 1994.

PIO98, S. J. Piola, Uso da Linguagem RS no Controle do Robô Nachi SC15F, Trabalho de Conclusão de Curso de Graduação, Departamento de Informática, UCS, Caxias do Sul, Brasil, 1998.

PMatos, P. J. Matos, Estudo e Desenvolvimento de Sistemas de Geração de Back-ends do Processo de Compilação, Dpto de Informática, Universidade do Minho, 1999, Masters Dissertation, Braga, Portugal.

PRO95, T. A. Proebsting, BURS Automata Generation, ACM Transactions on Programming Languages and Systems, 1995, v 17, nº 3, p 461-486.

REH97, J. A. Rehg, Introduction to Robotics in CIM Systems, Prentice Hall, Inc., 1997, 3rd edition.

RF95a, N. Ransey and M. Fernandez, New Jersey Machine-code Toolkit, Departament of Computer Science, Princeton University, 1995, Technical Report.

RF95b, N. Ransey and M. Fernandez, The New Jersey Machine-code Toolkit, Proceedings of The USENIX Technical Conference, 1995, p 289 - 302, New Orleans.

RF96, N. Ransey and M. Fernandez, New Jersey Machine-code Toolkit Architecture Specifications, Departament of Computer Science, Princeton University, 1996, Technical Report.

Sebesta99, R. W. Sebesta, Concepts of Programming Languages. Addison Wesley Longman Inc., 4th edition, 1999.

STA00, R. Stallman, Using and Porting the GNU Compiler Collection (GCC), iUniverse.com, Inc, 2000.

Tay82, R. H. Taylor and P. D. Summers and J. M. Meyer, AML: a manufacturing language, The International Journal of Robotics Research, 1982, v 1, nº 3.

Toscani93, S. S. Toscani, RS: Uma Linguagem para Programação de Núcleos Reactivos, Depto de Informática, UNL, 1993, Tese de doutoramento, Lisboa, Portugal.