Ambiente multiplataforma para o processo de ensino-aprendizagem de programação de microcontroladores

Leonardo Batista Moreira[1], Rodrigo Filev Maia[1]

[1]Departamento de Ciência da Computação – Centro Universitário da FEI (FEI)
Caixa Postal 09850-901 – São Bernardo do Campo – SP – Brasil

leonardo.moreira1987@hotmail.com, rfilev@fei.edu.br

Abstract. *This paper presents an ongoing work which proposes an approach learning programming for microcontrollers through a wizard whom helps the student-programmer to develop code for specific microcontroller platforms. The wizard helps students providing templates of codes for several microcontroller platforms, starting that each one can contain peculiarity of the architecture. The results intend that wizard may assist students in the learning process by providing codes and explanations about the functionality of the microcontroller under use by students.*

Resumo. *Este trabalho em desenvolvimento apresenta uma abordagem no processo de ensino-aprendizagem de programação de microcontroladores através de um wizard que auxilie o estudante a inserir funcionalidades em um projeto de software para uma determinada plataforma de microcontrolador. Tal wizard auxilia de acordo com cada template de microcontrolador, sendo que cada um pode conter peculiaridade da arquitetura e família de microcontroladores. Os resultados esperados são que o wizard auxilie no aprendizado propondo códigos e fornecendo explicações sobre a utilização da funcionalidade do microcontrolador representada pelo código fornecido pelo wizard durante sua utilização.*

1. Introdução

Atualmente existem diversas plataformas dedicadas ao desenvolvimento de *software*, sendo que apenas uma parcela destas plataformas é especializada ou possuiu ferramentas para desenvolvimento de aplicações para microcontroladores. Embora tais plataformas sejam imprescindíveis, as mesmas não oferecem auxílio para a produção de códigos para diferentes plataformas de microcotroladores, sendo que o desenvolvedor deve se preocupar com os detalhes de cada arquitetura de *hardware* sem auxílio da plataforma de desenvolvimento.

Uma das plataformas mais utilizadas atualmente para o desenvolvimento de aplicações denomina-se Eclipse, sendo composta por diversos módulos e ferramentas que, dentre outras, permite a criação de *plugins*, ou seja, de outras ferramentas acessórias ao desenvolvimento de aplicações. Torna-se, portanto possível elaborar na plataforma Eclipse uma ferramenta ou um conjunto de ferramentas que auxiliem o desenvolvedor a elaborar códigos para diferentes arquiteturas de *hardware*.

Para a elaboração de ferramentas que auxiliem o desenvolvedor na elaboração de aplicações para microcontroladores é utilizada a tecnologia Eclipse RCP que faz reuso de *plugins* do Eclipse para que possa ser rapidamente desenvolvida uma nova plataforma. Para esses objetivos o ambiente de desenvolvimento (IDE) tem como base um compilador de código aberto para compilação e montagem das aplicações programadas no ambiente proposto, assim como a IDE oferecerá *wizards* de programação que auxiliem o desenvolvedor a explorar os recursos de diferentes templates de microcontroladores. Tais wizards auxiliam o programador a identificar conjuntos de instruções válidas para um conjunto determinado de template podendo assim agregar tal funcionalidade ao projeto.

2. A plataforma Eclipse

O Eclipse IDE é uma plataforma de desenvolvimento gratuita e *open-source*, projetada para construção de ferramentas e ambientes de desenvolvimento integrados não contendo dependência de sistema operacional. Para que se possa garantir flexibilidade, a plataforma segue uma arquitetura que contêm classes e interfaces bem definidas garantindo a extensão e o suporte para o desenvolvimento de possíveis ferramentas [Org 2010].

2.1. Eclipse *plugin*

A menor unidade funcional do Eclipse são os *plugins*, que com exceção do núcleo da plataforma, compõem a totalidade do projeto. Um *plugin* tem por característica ser um modulo pequeno e leve usado para agregar funções a outros programas que são teoricamente maiores, adicionando alguma funcionalidade que seja especial ou que seja específica. Um *plugin* é programado em Java na plataforma Eclipse dentro do ambiente de desenvolvimento denominado *Plug-in Development Environment* (PDE) e o modelo de extensão existente baseia-se na criação de pontos de extensão para tais unidades funcionais, garantindo assim a interconectividade entre os módulos contidos na plataforma Eclipse [International 2003].

2.2. Eclipse PDE

O PDE é um ambiente de desenvolvimento de *plugin* contida na arquitetura da plataforma Eclipse, que provê ferramentas para auxiliar os desenvolvedores na criação, desenvolvimento, teste, depuração, compilação e implantação de *plugins* de modo ágil e fácil [Pde 2010].

2.3. Eclipse RCP

Originalmente, a plataforma Eclipse foi projeta para funcionar como uma plataforma de ferramentas abertas, no entanto o Eclipse a partir da versão V3.0 teve sua arquitetura reformulada de modo que seus componentes pudessem ser utilizados para construir praticamente qualquer aplicação cliente. O conjunto mínimo de *plugins* necessários para construir uma aplicação *rich client* é coletivamente conhecido como RCP (*Rich Client Platform*), sendo assim, RCP é um conceito que surgiu para facilitar a criação de novas aplicações em Java, com grande riqueza visual em *interfaces*, *layouts* e *menus*, porém de uma forma independente de plataforma [Minocha 2006].

O RCP utiliza SWT e Jface para desenvolvimento das aplicações. SWT (*Standard Widget Toolkit*) é a ferramenta gráfica utilizada pelo Eclipse no desenvolvimento e aplicações [Silva 2009]. Jface é um conjunto de classes que servem como apoio para o desenvolvimento de aplicações SWT no Eclipse [Junior 2006].

3. Wizard

Neste projeto foi desenvolvido um *wizard* para auxílio aos desenvolvedores de aplicações para microcontroladores. O *wizard* funciona de modo que para cada modelo de micontrolador seja elaborado um *template* composto por códigos específicos que podem ser utilizados no código fonte do estudante-desenvolvedor em forma de comentário podendo o estudante-desenvolvedor utilizá-los para executar algumas funcionalidades adicionais, na qual terá a necessidade de se alterar pouco código para se adaptar a aplicação que está sendo desenvolvida.

O *wizard* segue um fluxo que está descrito na Figura 1.

Figura 1: Fluxo do wizard

A Figura 1 descreve a ordem cronológica dos passos efetuados pelo *wizard*, sendo que primeiramente é verificado o arquivo com formato XML e o arquivo que neste momento está no padrão é verificado os nós procurando por pontos de extensão de código, havendo pontos de extensão eles são solicitados e aparecem no decorrer do *wizard* para que o usuário possa decidir se a funcionalidade é importante para aquele projeto e assim marcar para que o código possa ser agregado no *template* do projeto.

3.1. Arquitetura

Para cada *template* de microcontrolador do projeto deverá conter um arquivo no formato XML com o nome wizard.xml no diretório padrão do *template*, esse arquivo contém a definição do código dos pontos de extensão em sua estrutura. O arquivo deve ser lido e

interpretado pelo *wizard* de modo que no decorrer das ações exercidas pelo usuário ele possa decidir se deseja agregar ao projeto tal funcionalidade.

O arquivo wizard.xml deve seguir uma estrutura básica necessária para se tornar válido, tal estrutura pode ser visualizada na Figura 2.

```
<?xml version="1.0" encoding="utf-8" ?>
<wizard>
  <item>
    <page>/template/exemploFuncionalidade.c</page>
    <tag>// |tagExemplo|</tag>
    <code>
      <![CDATA[
          // Exemplo de texto para // funcionalidade que pode
          // ser agregada no projeto // no arquivo exemploFuncionalidade.c
      ]]>
    </code>
  </item>
</wizard>
```

Figura 2: Estrutura básica do wizard

A Figura 2 mostrou a estrutura básica necessária para o arquivo se tornar válido, sendo que ele deve conter um nó *root* denominado *wizard*, e para cada nó *item* deve conter os seguintes nós: *page*, *tag* e *code*.

O nó *page* deve conter o caminho relativo do arquivo que será alterado, de modo que ele é necessário para se saber exatamente para qual arquivo da estrutura de arquivos de programação será adicionado funcionalidades. O nó *tag* é uma cópia do componente que foi marcado no arquivo selecionado por *page*, na qual será localizado essa *tag* e haverá um substituição da *tag* pelo código contido no nó *code*. O nó *code* contém o bloco de código que será substituido no lugar da *tag* contida no arquivo.

4. Análise de Resultados

No *template* do projeto existem arquivos marcados, esses arquivos representam o lugar onde será substituído o texto como pode ser visto na Figura 3, o texto a ser substituído encontra-se no arquivo wizard.xml como pode ser visto na Figura 2.

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
  // |tagExemplo|

  printf("exemplo de codigo a ser adicionado\n");
  system("PAUSE");
  return 0;
}
```

Figura 3: Exemplo de código marcado

O arquivo wizard.xml é lido pelo *wizard* preenchendo os campos da aplicação como pode ser visto na Figura 4, esse conteúdo pode ser agregado ao código do projeto dependendo da opção do desenvolvedor. O conteúdo do *wizard* é substituído na marcação contida na Figura 3.

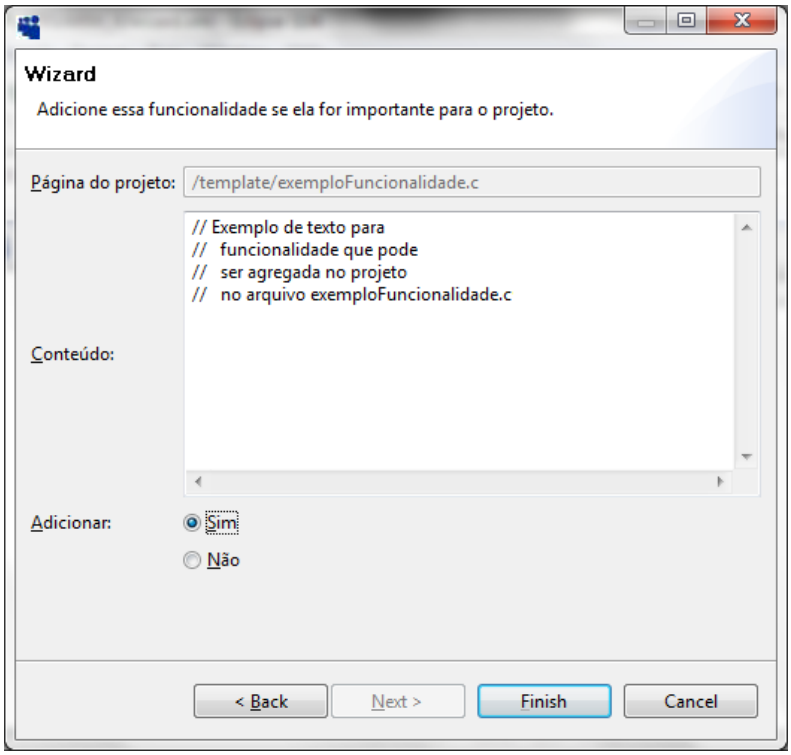

Figura 4: Wizard

O arquivo do *template* é alterado com o conteúdo marcado, adicionando assim a funcionalidade do microcontrolador a ser utilizada posteriormente pelo desenvolvedor como pode ser visto na Figura 5.

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
  // Exemplo de texto para
  //   funcionalidade que pode
  //   ser agregada no projeto
  //   no arquivo exemploFuncionalidade.cpp

  printf("exemplo de codigo a ser adicionado\n");
  system("PAUSE");
  return 0;
}
```

Figura 5: Exemplo de código alterado

5. Considerações Finais

Este projeto em andamento tem como resultados obtidos que um *wizard* pode ser um elemento de auxílio na montagem de códigos para plataformas complexas e que não se enquadram no que o estudante de um modo geral está familiarizado. O ensino de microcontroladores na engenharia e na ciência da computação é complexo e desse modo pode-se mostrar ao estudante como programar em tais plataformas, pois um dos requisitos de tais ambientes foi desenvolver a aplicação de acordo com os requisitos da estrutura do microcontrolador. Sendo assim, esta metodologia de ensino ajuda ao desenvolvedor que necessita utilizar algumas funcionalidades avançadas do microcontrolador.

References

International, O. T. (2003). "Eclipse Platform Technical Overview". IBM Corporation, http://www.eclipse.org/whitepapers/eclipse-overview.pdf, Junho.

Junior, M. L. A. (2006) "SWT, JFace e Componentes" http://www.devmedia.com.br/articles/viewcomp.asp?comp=3255, Maio.

McAffer, J. Lemieux, J.M. Aniszczyk C. (2010) Eclipse Rich Client Platform, Second edition. EclipseRT, May.

Minocha, S., "WebSphere JumpStart ISV Enablement", IBM Software Group, IBM Toronto Lab 27/Jun/2006http://www.ibm.com/developerworks/br/library/os-ecl-rcpapp/index.html

Silva, V. (2009) Practical Eclipse Rich Client Platform Projects. Apress.

Pde, The Eclipse Foundation (2010). "Plug-in development environment (PDE)". http://www.eclipse.org/pde.

Org, The Eclipse Foundation (2010). "About the Eclipse Foundation". http://www.eclipse.org/org/.