POO COM Tcl/Tk NA AUTOMAÇÃO DE LABORATÓRIO

Professor Jose

prof_jose@c2o.pro.br

Sumário: O objetivo deste trabalho é mostrar o uso de Software Livre para reduzir os custos e facilitar o acesso a recursos de automação para aquisição e armazenamento de dados gerados em laboratórios de *pesquisa*. Como ferramenta de software foi utilizada a linguagem Tcl/Tk pelas suas características de extensibilidade e portabilidade utilizando o paradigma da Programação Orientada a Objetos (POO).

Palavras-chave: Automação, Software Livre, POO

1. INTRODUÇÃO

A aquisição e armazenamento de dados experimentais aumenta o volume de dados e elimina os erros associados ao registro e à transcrição manual dos mesmos. Essa é uma necessidade de profissionais de diferentes áreas que em algum momento do seu trabalho necessitam monitorar e/ou controlar uma montagem experimental provisória de um sistema químico, biológico ou físico.

Grande parte dos equipamentos (bombas peristálticas, válvulas de injeção e válvulas multiporta, espectrofotômetros, potenciômetros, balanças, dentre outros) necessários para automação já saem de fábrica com recursos para comunicação serial, através de comandos em código ASCII e portanto em condições de se comunicar com um PC através da porta serial. Portanto muitos laboratórios já dispõem de recursos físicos (*Hardware*) para automatizar procedimentos. [1]

Os programas de monitoramento/controle podem ser desenvolvidos, de maneira simplificada, com ferramentas de Software Livre tais como a linguagem Tcl/Tk.

A Tcl (*Tool Command Language*) é uma linguagem de programação multiplataforma de fácil aprendizagem, em comparação com outras linguagens, mas com muitos recursos. A Tcl pode ser utilizada para a escrita de programas com interface gráfica usando a extensão Tk, a biblioteca gráfica padrão para a Tcl. [2-4]

A extensão Metakit, escrita em C^{++} oferece uma API para Tcl (Mk4tcl) facilitando significativamente o desenvolvimento de aplicações para armazenamento e gerenciamento de dados estruturados. [5]

1.1 Programação Orientada para Objetos (POO)

Através da filosofia da POO um programa é concebido como uma *coleção de objetos* ao invés de uma lista de instruções, como na programação tradicional. Esses *objetos* vêm associados com as próprias operações que eles realizam, chamadas de *métodos*, na terminologia da POO. Os *métodos* portanto, servem para fazer os *objetos* operarem algo. Além dos *métodos* os *objetos* também possuem *atributos* que são propriedades caracterizando a estrutura de um dado objeto. Os *objetos*, nesse tipo de programação, são organizados em *classes* hierárquicas as quais descrevem um ou mais *objetos* similares. [6]

1.2 POO com Tcl/Tk

C^{++} e Java são linguagens bem conhecidas que oferecem suporte nativo para POO. Isso não significa que POO não seja possível em outras linguagens, pois POO é uma maneira de pensar, de modelar o sistema. Tem mais a ver com a análise e modelagem do sistema do que com a sua implementação. A Tcl não oferece tipos primitivos para POO, mas a sua flexibilidade permite a criação de primitivas para a criação de objetos. [7]

Existem também extensões que facilitam o uso da POO com Tcl, tais como: Incr Tcl, Stooop e OTcl. [8]

2. POO NA AUTOMAÇÃO DE LABORATÓRIO

É possível identificar dois objetos típicos na Automação de Laboratório para serem modelados segundo a POO, os *equipamentos* e os *instrumentos*. Um *instrumento* poderia ser definido como um dispositivo que permite realizar medições e portanto fornece algum tipo de informação (qualitativa ou quantitativa), enquanto um *equipamento* serve para realizar alguma tarefa sem o compromisso de gerar informações sobre o sistema que se deseja estudar. Assim, pHmetro é um *instrumento*, enquanto uma bomba peristáltica é um *equipamento*. Esses objetos fariam parte de uma classe mais geral *Hardware Analítico*, conforme o diagrama de classes na figura 1. [9]

Portanto a classe instrumento possui os seguintes atributos: unidade, estado e leitura. O atributo unidade armazena a unidade de leitura (Ex: mg/L, mV), o estado identifica o estado atual do instrumento (monitorando, parado, pausado, calibrando ou condicionando) e a leitura armazena a última leitura realizada como uma lista no formato {tempo, leitura}.

A classe *Instrumento* pode representar um sensor de temperatura, um eletrodo íon-seletivo ou um sistema de análise completo.

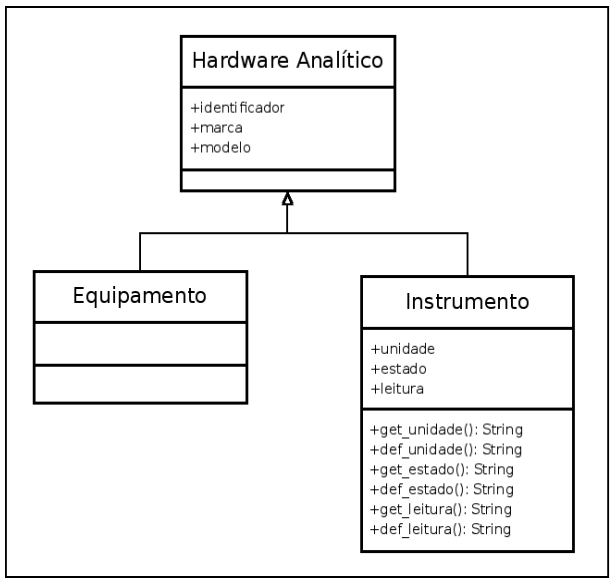

Fig. 1. Diagrama de classe para a especialização da classe Hardware Analítico nas classes Equipamento e Instrumento. [9]

2.1 POO no Gerenciamento de Dados

A automação da aquisição de dados aumenta significativamente o volume de dados para serem analisados, ainda que o número de amostras ou parâmetros seja pequeno. Em um contexto de pesquisa, a análise do conjunto de dados de um projeto é um suporte importante para tomada de decisões, por isso foram modeladas a classe *Amostra* e *Medida* conforme o diagrama de classes da figura 2.

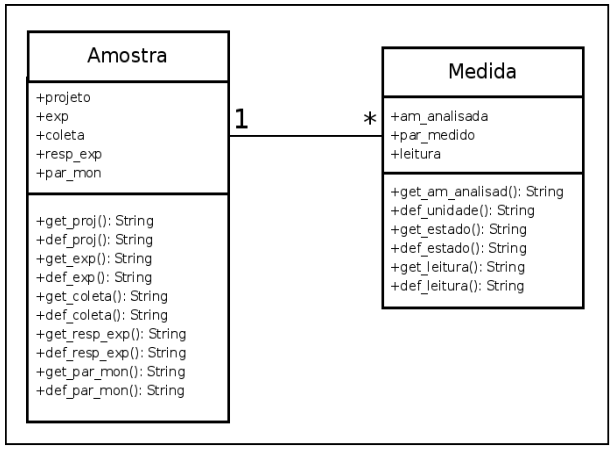

Fig. 2. Diagrama das classes Amostra e Medida para o gerenciamento de dados experimentais.

Durante um projeto de pesquisa são realizados vários experimentos durante os quais, podem ser usadas amostras únicas (coleta manual em campo) ou várias amostras (coleta automática), por isso a classe A*mostra* possui os seguintes atributos: projeto, experimento (exp), coleta, responsável pelo experimento (resp_exp) e parâmetros monitorados (par_mon).

Os atributos dos objetos da classe Amostra armazenam as seguintes informações:

-projeto {codigo coordenador}

-experimento {numero estado_atual}

A variável "estado_atual" guarda o estado do experimento que estiver em andamento e pode assumir os seguintes estados: iniciando, andamento, pausado interrompido ou realizado.

-coleta {local data}

-responsavel_experimento

-parametros_monitorados {p1 lim_inf lim_sup ...}

A variável p1 armazena o nome do parâmetro monitorado e está associada às variáveis lim_inf e lim_sup que representam respectivamente o limite inferior e superior. A definição desses limites permite executar ações pré-definidas quando os parâmetros monitorados estiverem fora dos limites estabelecidos.

A multiplicidade "1" e "*" na associação entre Amostra e Medida, significa que um objeto da classe Amostra pode estar associado a vários objetos da classe Medida para diferentes parâmetros em momentos diferentes, mas um objeto da classe Medida deve estar associado a apenas um único objeto da classe Amostra.

O atributo am_analisada (amostra analisada) da classe *Medida*, permite discriminar diferentes locais de amostragem durante um experimento, por exemplo entrada e saída de um biorreator, simplificando a implantação de um sistema da amostragem automatizado para monitoramento contínuo de uma montagem experimental.

As classes *Instrumento*, *Amostra* e *Medida* estão associadas entre si e com o "Banco de Dados" no contexto de um "Experimento" conforme o diagrama da figura 3.

Um instrumento pode estar monitorando uma amostra de cada vez, gerando uma leitura e permitindo a criação de uma instância da classe *Medida.*

Uma única amostra pode ser monitorada por vários instrumentos gerando várias medidas, mas uma instância da classe *Medida* só poderá estar associada a um único instrumento e uma única amostra.

Quando um dado chegar pela porta serial, será verificado se o objeto instrumento existe "e" se está no estado monitorando para a criação de uma instância da classe *Medida* que irá consolidar todas as informações (dados e meta-dados) dos respectivos objetos instrumento e amostra para armazenamento no banco de dados.

No caso de existirem diversos locais de medida para uma mesma amostra, então o instrumento deve permanecer no estado “condicionando” para a troca de amostras e neste caso, a leitura não seria armazenada.

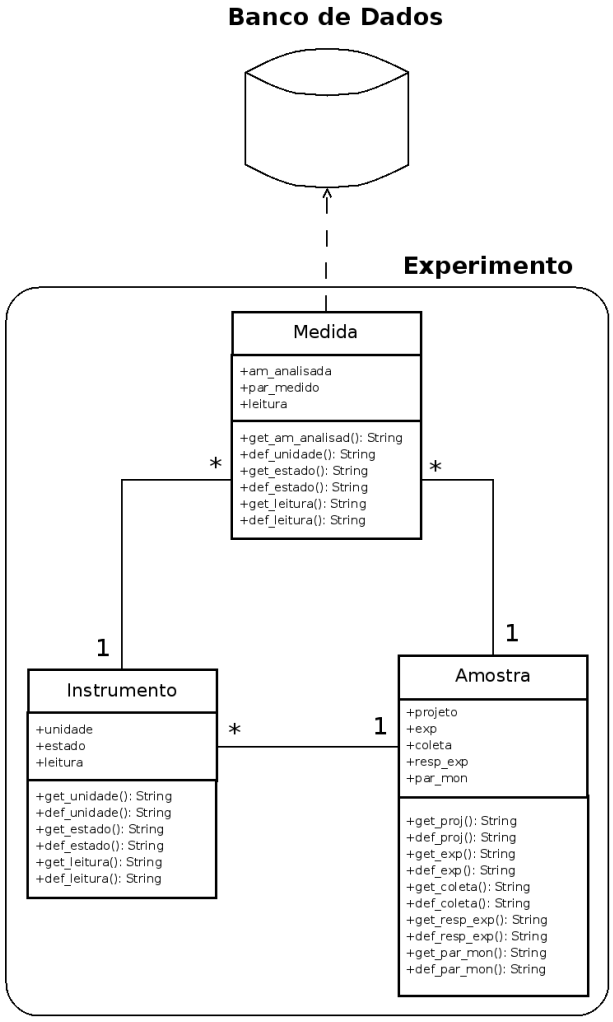

Fig. 3. Diagrama das associações entre as classes Instrumento, Amostra e Medida com o “Banco de Dados” no contexto de um “Experimento”.

3. UMA APLICAÇÃO PRÁTICA DA POO COM Tcl/Tk NA AUTOMAÇÃO DE LABORATÓRIO

O modelo orientado a objetos foi implementado com a linguagem Tcl/Tk no desenvolvimento de um programa (GPL) para automatizar a aquisição e o gerenciamento de dados, gerados por um potenciômetro multiparâmetro utilizado no nosso laboratório em montagens experimentais para estudos de tratabilidade de efluentes industriais. [10]

O equipamento utilizado permite a leitura simultânea de até 3 eletrodos (pH ou ISE, Condutividade e Oxigênio Dissolvido) e pode ser configurado para enviar, pela porta serial RS-232, as informações exibidas no display em intervalos de tempo pré-definidos. Essas informações são codificadas no formato ASCII e podem ser processadas e armazenadas em um computador.

Os atributos são armazenados em variáveis do tipo array com escopo global, as quais são atualizadas pelos respectivos métodos para atualizar (def_...) e resgatar o valor atual (get_...).

Por exemplo, um objeto da classe Instrumento possui a unidade de medida como um dos atributos, o qual é um array global e será indexado pelo nome do objeto e armazena como valor a unidade de concentração.

O método “get_unidade” é implementado em Tcl da seguinte forma:

```
proc get_unidade { nome_sensor } {

   global unidade

   if { [info exists unidade($nome_sensor)] } {

        return $unidade($nome_sensor)

   } else {

        puts "Atenção: $nome_sensor não tem unidade"
        return "unidade_indefinida"

   }
}
```

E o método def_unidade:

```
proc def_unidade { nome_sensor valor_unidade } {

global unidade

set unidade($nome_sensor) $valor_unidade

}
```

A criação de um “objeto” da classe *Instrumento* foi implementada com dois procedimentos, “cria_instrumento” e “instrumento”, descritos a seguir:

```
proc cria_instrumento { nome_instrumento comando args } {
   if { $comando == "get_unidade" } {
         return [get_unidade $nome_instrumento]
      } elseif { $comando == "def_unidade"} {
         def_unidade $nome_instrumento [lindex $args 0]
      } elseif { $comando == "get_estado_instrumento" } {
         return [get_estado_instrumento $nome_instrumento]
      } elseif { $comando == "def_estado_instrumento"} {
         def_estado_instrumento $nome_instrumento [lindex $args 0]
      } elseif { $comando == "get_leitura" } {
         return [get_leitura $nome_instrumento]
      } elseif { $comando == "def_leitura"} {
         def_leitura $nome_instrumento [lindex $args 0]
      } else {
         puts "Erro: $comando é um comando desconhecido"
      }
}

proc instrumento {args} {
   foreach nome $args {
        proc $nome {comando args} "return \[eval cria_instrumento
$nome \$comando \$args\]"
   }
}
```

O argumento "args" é um nome especial que armazena

todos os argumentos restantes em uma lista de tamanho variável.

A criação de “objetos” utiliza espaços de memória que precisam ser liberados quando não forem mais necessários, por isso a necessidade de uma rotina para liberar recursos chamada remover_instrumento:

```
proc remover_instrumento { args } {

   global unidade
   global estado_instrumento
   global leitura

   foreach nome $args {
      if { [info exists unidade($nome)] } {
                  unset unidade($nome)  ;#Deleta os dados do objeto
         }
         if { [info exists estado_instrumento($nome)] } {
                     unset estado_instrumento($nome)     ;#Deleta os
dados do objeto
                  }
                  if { [info exists leitura($nome)] } {
                              unset leitura($nome) ;#Deleta os dados
do objeto
                   }
                   rename $nome {}      ;#Deleta o comando objeto
   }
}
```

Procedimentos análogos foram codificados para a criação de objetos das demais classes.

3.1 Interface Gráfica

O programa desenvolvido exibe uma tela inicial de configuração que permite ao usuário selecionar os eletrodos instalados, as unidades de medida e a porta serial em que está conectado o equipamento, conforme mostra a figura 4.

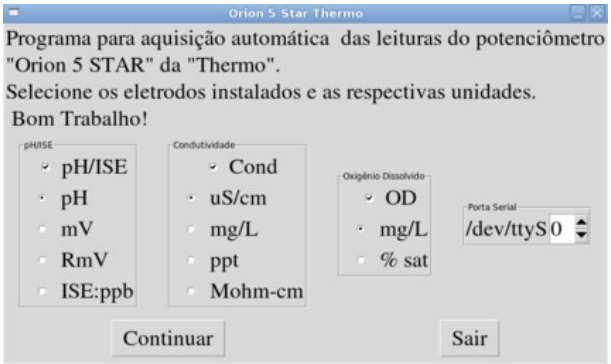

Fig. 4. Tela inicial de configuração do programa de aquisição de dados para o potenciômetro multiparâmetros.

Ao clicar no botão “Continuar” o programa abre a tela de monitoramento que permite visualizar seletivamente as leituras dos eletrodos conectados, conforme a figura 5.

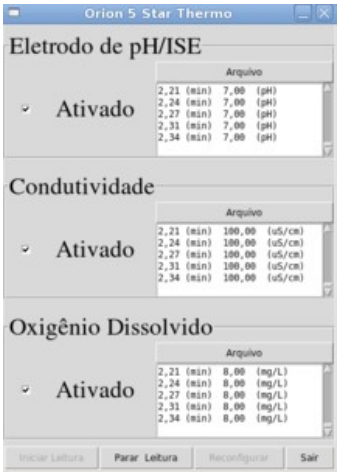

Fig. 5. Tela de monitoramento do programa de aquisição de dados para o potenciômetro multiparâmetros.

Os *radiobuttons* de cada eletrodo instalado ativa a criação ou remoção de objetos da classe *Instrumento.*

O botão “Iniciar Leitura” da tela de monitoramento inicia um contador de tempo, único para todas as leituras, e o registro dos pares de dados {tempo leitura}, simultaneamente abre a tela para cadastramento do experimento, figura 6.

Os dados registrados na tela de monitoramento podem ser salvos pelo usuário como um arquivo “CSV” para posterior edição por uma planilha de cálculo (Ex: Open Office Calc).

As informações fornecidas pelo usuário na tela de cadastramento são armazenadas nas seguintes variáveis:
cod_proj => código do projeto
coord_proj => coordenador do projeto
cod_exp => código do experimento
resp_exp => responsável pelo experimento
local_col => local da coleta
data_col => data da coleta (dia/mes/ano hora:min)
resp_col => responsável pela coleta

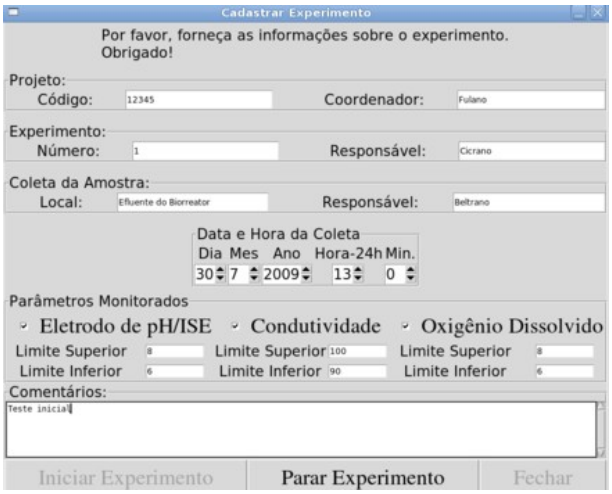

Fig. 6. Tela de cadastramento de experimentos do programa de aquisição de dados para o potenciômetro multiparâmetros.

O botão “Iniciar Experimento”, habilita a criação dos objetos das classes *Medida* e *Amostra* para o gerenciamento e armazenagem dos dados enviados pelo potenciômetro.

Portanto, o botão “Iniciar Leitura” apenas habilita o registro das leituras na tela de monitoramento, mas o armazenamento automático em banco de dados somente ocorre com o início de um experimento.

O gerenciamento do banco de dados está sendo implementado com a ferramenta “Metakit” que simplifica significativamente o armazenamento e resgate dos resultados experimentais, os quais são gravados em um arquivo binário na máquina local ou na rede.[5]

5. CONCLUSÕES

O paradigma da POO foi implementado usando os recursos nativos da linguagem Tcl, oferecendo como principal vantagem a reutilização de código o que facilita a inclusão de novos recursos ao programa com pouco esforço de programação.

Ferramentas de Software Livre (GPL) estão sendo utilizadas com sucesso no desenvolvimento de programas para aquisição e gerenciamento de dados experimentais, colaborando para aumentar a produtividade, qualidade e a rastreabilidade dos resultados de análises químicas e montagens experimentais ao longo de projetos de pesquisa.

Isso mostra que soluções “abertas” (GPL) podem contribuir para aumentar o índice de automação em laboratórios.

REFERÊNCIAS

[1] http://www.c2o.pro.br/automacao/x783.html, Acesso em: 30 de julho de 2009

[2] John K. Ousterhout, “Tcl and Tk toolkit: Addison-Wesley Professional Computing Series”, Addison-Wesley, 1994

[3] Mark Harrison e Michael McLennan, “Effective Tcl/Tk Programming: Addison-Wesley Professional Computing Series”, Addison-Wesley, 1998

[4] http://pt.wikipedia.org/wiki/Tcl, Acesso em: 30 julho 2009

[5] www.equi4.com/metakit/tcl.html, Acesso em 30 julho 2009

[6] Agenor Martins,, “O que é programação”. Editora Brasiliense S/A, 1996

[7] http://users.telenet.be/koen.vandamme1/papers/tcl_objects/tcl_objects.html, Acesso em: 30 julho 2009

[8] http://wiki.tcl.tk/970 , Acesso em: 30 julho 2009

[9] M. Urbano Cuadrado, M.D. Luque de Castro e M.A. Gómez-Nieto, “Object-oriented techniques for design and development of standard software solutions in automation and data management in analytical chemistry”, Trends Anal. Chem. Vol. 25, p 66, 2006

[10] http://www.thermo.com/com/cda/product/detail/0,1055,10121407,00.html , Acesso em: 30 julho 2009