VIII Seminário Técnico de Proteção e Controle
28 de Junho a 1º de Julho de 2005
Rio de Janeiro – RJ

Artigo no. ST-43

DESENVOLVIMENTO E IMPLEMENTAÇÃO DE DISPOSITIVOS DE PROTEÇÃO NO ATP

Renato Mikio Nakagomi[1*]
rmikio@pea.usp.br

Eduardo Cesar Senger[1]
senger@pea.usp.br

Giovanni Manassero Junior[1]
giomanjr@pea.usp.br

Eduardo Lorenzetti. Pellini[1]
epellini@pea.usp.br

Francisco Antônio Reis Filho[1]
mother@pea.usp.br

(1) Escola Politécnica da Universidade de São Paulo – EPUSP

1. RESUMO

Este artigo apresenta de forma detalhada como desenvolver e implementar algorítmos de função de proteção no ambiente ATP (*Alternative Transient Program*), utilizando funções programadas em linguagem C (denominadas *foreign models*). Demonstra-se neste trabalho a eficiência e eficácia desta abordagem frente ao tradicional uso das MODELS do ATP, pois o próprio código-fonte do algorítmo de funções de proteção desenvolvido para a implementação no *firmware* do relé pode ser utilizado como *foreign models* dentro do ambiente ATP. O trabalho desenvolvido acrescenta conhecimento a uma forma pouco explorada da utilização dos modelos no ATP. Especificamente para demonstração deste trabalho foi implementado a função 21 de um relé de distância em linguagem C.

Palavras-chave: simulação de sistemas de proteção, desenvolvimento de algorítmo, relé de distância, EMTP/ATP.

2. CONSIDERAÇÕES GERAIS

Com a introdução da eletrônica digital, os relés de proteção tornaram-se equipamentos bastante complexos e sofisticados que englobam diversas funções em uma única unidade (funções de proteção, registro de eventos, oscilografia, localização da falta, etc).

O trabalho apresentado neste artigo é parte da criação de um protótipo de um relé de distância apresentado em [1]. Segundo este estudo, os principais módulos de *software* existentes nesse equipamento são:

a) Módulo ANSI-21: responsável pela implementação da função de proteção de distância. É constituído pelos seguintes sub-módulos:
   a.1) Classificação da falta: utilizado para selecionar quais elementos de medição serão utilizados no cálculo da impedância durante cada particular condição operativa da rede.
   a.2) Cálculo da Impedância: responsável por calcular a impedância vista pelo relé utilizando o método da equação diferencial.
   a.3) Zonas características: recebe as impedâncias calculadas no módulo anterior e testa a condição de trip para as diversas zonas características de proteção. O relé utiliza, tanto para a proteção de faltas entre fases (função 21) quanto para a proteção de faltas fase-terra (função 21N), até 4 zonas com característica MHO ou QUADRILATERAL, configuráveis pelo usuário.
b) Módulo Power Swing: monitora a ocorrência de oscilações eletromecânicas na rede de potência e implementa as funções 68 (power swing blocking) e 78 (power swing tripping).
c) Módulo Lógica de Teleproteção: implementa as diversas lógicas disponíveis em teleproteção (POTT, PUTT, etc.).
d) Outras Funções de proteção: implementa todas as outras funções ANSI que estão normalmente disponibilizadas em um relé digital de distância (p. ex. 67/67N; 50/51; 27; 59; 79; etc.)
e) Facilidades de Análise Pós-Falta: integra os módulos de *software* para análise do evento tais como: oscilografia; seqüência de eventos e localização da falta.
f) Módulo de Medição: rotinas de *software* responsáveis pela medição das principais grandezas elétricas de interesse (tensões

*Depto. de Engenharia de Energia e Automação Elétricas da Escola Politécnica da Universidade de São Paulo – EPUSP
Av. Prof. Luciano Gualberto, 158 – travessa 3 – Bloco A – sala A2-14
CEP: 05508-900 – São Paulo – SP

fase/linha; correntes; potências ativa e reativa; freqüência, etc.)

g) Facilidades de Monitoramento: rotinas de auto-teste; monitoramento do circuito dos TP's; monitoramento de disjuntor; monitoramento do circuito de trip; etc.

3. IMPLEMENTAÇÃO DE DISPOSITIVO DE PROTEÇÃO EM ATP

A fim de testar o desempenho dos algoritmos desenvolvidos, foi utilizado o programa de simulação denominado ATP (*Alternative Transient Program*) através dos *foreign models*.

3.1 O ATP

O ATP é um programa computacional de simulação de redes elétricas de potência, sendo um dos mais conhecidos e conceituados programas do gênero. Este programa, além de realizar simulações transitórias e em regime permanente de sistemas elétricos, permite que o usuário crie seus próprios modelos de dispositivos de proteção para serem testados juntamente à simulação da rede. Isto pode ser feito de duas maneiras: a primeira refere-se à criação dos MODELS utilizando uma linguagem de programação própria do ATP, similar às linguagens PASCAL ou MODULA-2, baseada na linguagem denominada FORTRAN [2]; a segunda está relacionada aos chamados *foreign models* (ou modelos externos, em tradução livre), que são programas escritos em linguagem C e podem ser inseridos dentro do código do próprio ATP [3]. É preciso salientar aqui que somente usuários autorizados podem obter os arquivos-objeto para a reconstrução do núcleo do ATP com novas funcionalidades implementadas em linguagem C.

Permitir que a linguagem C seja usada para o desenvolvimento de programas e funções no ATP é uma grande vantagem, pois tal linguagem é amplamente divulgada e conhecida atualmente, ao contrário da linguagem de programação inerente ao ATP. Além disso, os programas e funções construídos a partir da linguagem C são mais rapidamente executados, pois são códigos compilados, otimizando o rendimento das simulações realizadas no próprio ATP.

Contudo, a maior vantagem em utilizar a linguagem C para descrever funções no ATP é a praticidade do desenvolvedor em implementar todos os algoritmos de proteção do relé diretamente no ATP através do *foreign model*, ou seja, o próprio código-fonte do algoritmo de funções de proteção a ser implementado no *firmware* de um relé pode ser aproveitado para a utilização no ambiente ATP como *foreign model*.

3.2 Criando *foreign models*

Um *foreign model* é basicamente um programa em C capaz de interagir com os sinais de tensão e corrente gerados durante a simulação de qualquer caso no ATP. A fim de utilizar um programa em C como um *foreign model*, é necessário adaptar o programa para que algumas poucas condições lógicas sejam respeitadas e depois criar um novo arquivo executável do ATP que, além do conteúdo original inerente ao algoritmo do ATP, possua as informações referentes às novas funcionalidades desenvolvidas. As condições a serem respeitadas durante a codificação de um *foreign model* são:

a) Caso se queira desenvolver um elemento a ser utilizado também com o ATPDRAW, é preciso que a função tenha no máximo 12 portas de *I/O* (ou seja, uma combinação de 12 portas de entradas e saídas), a fim de respeitar a dimensão limitada do ícone no ATPDRAW. Se o elemento implementado possuir mais do que 12 portas, o usuário não poderá se beneficiar da interface gráfica do ATPDRAW.

b) O programa em C não deve ser dividido em pequenos programas e bibliotecas. É melhor manter todas as funções e definições dentro do mesmo arquivo se possível.

c) O programa em C não deve possuir uma função `main()`. Para a implementação em ATP deverão ser criadas duas funções principais (nomeadas mnemonicamente se possível). A primeira será uma função de inicialização e a outra será a função de execução. A função de inicialização é executada somente uma vez no começo da simulação. A função de execução é rodada para cada passo de integração da simulação previamente configurado no ATP.

d) Ambas as funções mencionadas acima devem ser declaradas a fim de permitir o transporte dos parâmetros e sinais entre a simulação e as novas funções. Nestas declarações devem existir as seguintes variáveis declaradas como vetores do tipo double: `xdata_ar[]`, `xin_ar[]`, `xout_ar[]`, `xvar_ar[]`.

d.1) `xdata_ar[]` está relacionada ao vetor de variáveis que guardam valores fixos que não mudam durante a simulação, atuando somente como sinalizadores ou parâmetros de configuração/inicialização.

d.2) `xin_ar[]` está relacionada ao vetor de variáveis de entrada da função implementada. Estas variáveis guardam os valores dos sinais fornecidos pela simulação.

d.3) `xout_ar[]` está relacionada ao vetor de variáveis de saída da função implementada. Estas variáveis possuem os valores resultantes dos cálculos realizados dentro da nova funcionalidade que serão passados para a simulação.

Nota: Como mencionado anteriormente, se o usuário quiser os benefícios de uma interface gráfica (ATPDRAW), o número máximo de portas de entrada/saída é 12.

d.4) `xvar_ar[]` está relacionada ao vetor de variáveis locais da nova função implementada. Foi verificado que o valor mínimo a ser declarado como `xvar_ar[]` deve ser o mesmo

da quantidade de portas de saída existentes no *foreign model*.

3.3 Criando um arquivo executável do ATP personalizado

Depois de construído o programa em C relativo ao algoritmo que se desejava, é necessário incluí-lo no arquivo executável do ATP. Para tanto, é necessário compilar um novo arquivo executável do ATP contendo estas novas funções desenvolvidas em linguagem C. Primeiramente, é preciso alterar dois arquivos importantes que permitem reconstruir o núcleo do ATP: o FGNMOD.FOR e o OBJLIST.WNT.

O arquivo FGNMOD.FOR deve conter todas as declarações das funções adicionais desenvolvidas como foreign models. Neste caso, as alterações feitas foram a inclusão da função *RELAY21* como *refnam* e a inclusão das chamadas para as funções de inicialização e execução desta função (conforme mencionado anteriormente). Observe a formatação dos dados para a inclusão da função denominada *RELAY21* no quadro abaixo:

```
(...)
DATA refnam(2) / 'RELAY21' /
(...)
 ELSE IF ( iname.EQ.2 ) THEN
  IF (iniflg.EQ.1) THEN
   CALL relay21_i(xdata,xin,xout,xvar)
  ELSE
   CALL relay21_m(xdata,xin,xout,xvar)
  ENDIF
  CONTINUE !
(...)
```

Note que são declaradas duas funções: *relay21_i* e *relay21_m*, respectivamente as funções de inicialização e execução da nova funcionalidade implementada.

Por sua vez, o arquivo OBJLIST.WNT contém os nomes de todos os arquivos-objeto a serem compilados para gerar o novo executável do ATP, ou seja, para este caso é necessário somente incluir o nome dos novos arquivos-objeto relacionados com as novas funcionalidades. É importante observar que nenhum nome de arquivo deve ser alterado ou apagado do OBJLIST.WNT, uma vez que os demais arquivos-objeto declarados são obrigatórios por fazerem parte do núcleo funcional do ATP.

```
(...)
file relay21
```

Depois de preparar os arquivos FGNMOD.FOR e OBJLIST.WNT, basta compilar os programas desenvolvidos em C (já devidamente adaptados para foreign models), compilar o arquivo FGNMOD.FOR utilizando um compilador FORTRAN e depois executar o processo de link entre os arquivos-objeto do ATP e os recém-gerados arquivos-objeto das novas funcionalidades. No estudo atual, foram utilizados os compiladores para C e FORTRAN, wcc386 e wfc386 respectivamente, e o linker denominado wlink, todos pertencentes ao WATCOM 11.0 para Windows NT.

3.4 Utilizando a *foreign model*

Da mesma forma que se programam as MODELS tradicionais, é necessário descrever os cartões de *foreign models*.

As únicas diferenças existentes entre as declarações das MODELS e das *foreign models* são: a declaração da *foreign model* e da quantidade de variáveis utilizadas (**`ixdata, ixin, ixout e ixvar`**); e a presença da atribuição das variáveis `xdata[]`, `xin[]` e `xout[]`. Excetuando-se estes fatos, a descrição dos cartões de MODELS permanece inalterado.

Os cartões do ATP para *foreign models* possuem o seguinte formato:

```
MODELS
INPUT
   …
OUTPUT
   …
 MODEL relay21
  INPUT
   …
  DATA
   …
  OUTPUT
   …
  VAR
   …
  MODEL REL21 FOREIGN RELAY21
   {ixdata:32, ixin:7, ixout:3,
   ixvar:3}
   EXEC
    USE REL21 AS REL21
     DATA
      xdata[…]:=VARDATA
      …
     INPUT
      xin[…]:= VARIN
      …
     OUTPUT
      VAROUT:=xout[…]
      …
    ENDUSE
   ENDEXEC
 ENDMODEL
  USE RELAY21 AS RELAY21
  INPUT
    VARIN:= <nós do sistema>
    …
  DATA
    VARDATA:= <valores>
    …
  OUTPUT
    OUTP:=VAROUT
    …
  ENDUSE
ENDMODELS
```

Mais detalhadamente, a declaração da *foreign model* é feita como no exemplo descrito abaixo:

`MODEL relay21` → declaração do modelo. Este é o nome pelo qual o ATP fará referência.

`INPUT ...` → declaração das variáveis de entrada. Estes nomes devem ser guardados para futura referência na interface gráfica.

`DATA ...` → declaração dos parâmetros do modelo acessíveis através da configuração do elemento pelo ATPDraw.

`OUTPUT ...` → declaração das variáveis de saída. Estes nomes devem ser guardados para futura referência na interface gráfica.

`VAR ...` → declaração de variáveis de manipulação. Estas variáveis serão referenciadas internamente a princípio. É importante observar que as variáveis de saída (`OUTPUT`) também aparecem nesta declaração. Outras variáveis (que não sejam de saída) podem ser acrescentadas caso haja necessidade (p.ex. variáveis auxiliares).

`MODEL REL21 FOREIGN RELAY21 {ixdata:32, ixin:7, ixout:3, ixvar:3}` → declaração da chamada da função indicada anteriormente em FGNMOD.FOR associada a um nome fantasia. Note que os parâmetros desta função correspondem aos tamanhos dos vetores associados descritos anteriormente.

`EXEC` → comando para início de execução da chamada de função

`USE REL21 AS REL21` → chamada da função pelo nome fantasia. Este comando serve para distingüir vários modelos iguais utilizados num mesmo circuito de simulação.

`INPUT` → comando para definição da ordem e atribuição das variáveis de entrada.

```
xin[1]:=VARIN1
```

`DATA` → comando para definição da ordem e atribuição dos parâmetros do elemento.

```
xdata[1]:=VARDATA
```

`OUTPUT` → comando para definição da ordem e atribuição das variáveis de saída.

```
OUTP1:=xout[1]
```

`ENDUSE` → finalização do comando *USE*.

`ENDEXEC` → finalização do comando *EXEC*.

`ENDMODEL` → finalização do comando *MODEL*.

3.5 Criando o modelo no ATPDraw

Para facilitar a utilização de um MODEL ou *foreign model* é possível criar elementos no ATPDraw. No entanto, um ícone somente não basta para representar um modelo no ATPDraw. É preciso definir o modelo realmente, indicando o número, tipo e localização no próprio ícone das entradas e saídas que este modelo contempla.

Existem dois tipos de configurações a serem definidas no ATPDraw para tal propósito: a definição de especificação do usuário (*User Spec.*) e a definição do modelo em si (*Model*). Por motivos de limitação de espaço não será possível detalhar a criação de um elemento no ATPDraw passo a passo. Porém é importante destacar algumas regras a serem seguidas para tal criação:

a) Para criar um ícone no ATPDraw, primeiramente é necessário saber o número de entradas e de saídas que serão utilizadas, pois há uma limitação gráfica de 12 (doze) portas como citado anteriormente.
b) Antes de lidar com a parte gráfica, é necessário saber como a nova função deverá ser implementada num arquivo ATP através do ATPDraw. Para isso é necessário informar ao ATPDraw como é o modelo que está sendo criado, quais são as variáveis envolvidas de entrada, saída e manipulação e quais são as funções que deverão ser executadas pelo modelo. O arquivo .MOD nada mais é do que o conjunto de cartões utilizados em um arquivo .ATP que descrevem um MODEL ou um *foreign model*. Assim, similarmente ao descrito no item 3.4, tem-se dentro do arquivo .MOD:
c) Os nomes das variáveis utilizadas no programa não devem ultrapassar 6 (seis) caracteres, pois esta é uma limitação do próprio ATP.
d) Todos os arquivos criados durante o desenvolvimento do elemento para o ATPDraw devem possuir nome de até 8 (oito) caracteres, pois é uma limitação do ATPDraw para DOS.
e) Graficamente também existem somente 12 (doze) posições em que as portas (de entrada e saída) podem ser conectadas. Para que o modelo fique esteticamente correto, é preferível que sejam seguidas as marcações dos pinos das portas, visualizadas nas bordas do editor de ícones do ATPDraw para DOS.

4. CIRCUITO PARA SIMULAÇÃO

Através do ATPDraw para DOS, foi criado um circuito para testar a implementação da função 21 do relé de distância.

Para utilizar o elemento criado no ATPDraw, deve-se observar algumas peculiaridades próprias do ATP referentes às ligações a serem feitas no ícone:

a) as entradas de tensão do elemento podem ser ligadas diretamente em seus pontos de conexão a partir dos pontos de interesse do circuito.
b) as entradas de corrente para o elemento apresentado neste projeto devem seguir as especificações recomendadas pelo próprio ATP, ou seja, toda variável que depender da grandeza de corrente deverá obter os dados a partir de uma chave (*switch*) ou de um medidor de corrente (*Probe curr.*).
c) as saídas do elemento desenvolvido para este projeto devem simular o controle de disjuntores de uma linha de transmissão. Para que isso seja possível, foram utilizadas chaves (*switches*) do tipo TACS, dispositivos encontrados no próprio ATPDraw que podem ter sua abertura ou fechamento controlados por um sinal externo. Neste caso, o sinal externo é providenciado pelas saídas do elemento implementado.

Na Figura 1 a seguir é possível observar como as ligações foram feitas e como o elemento *RELAY21* foi utilizado.

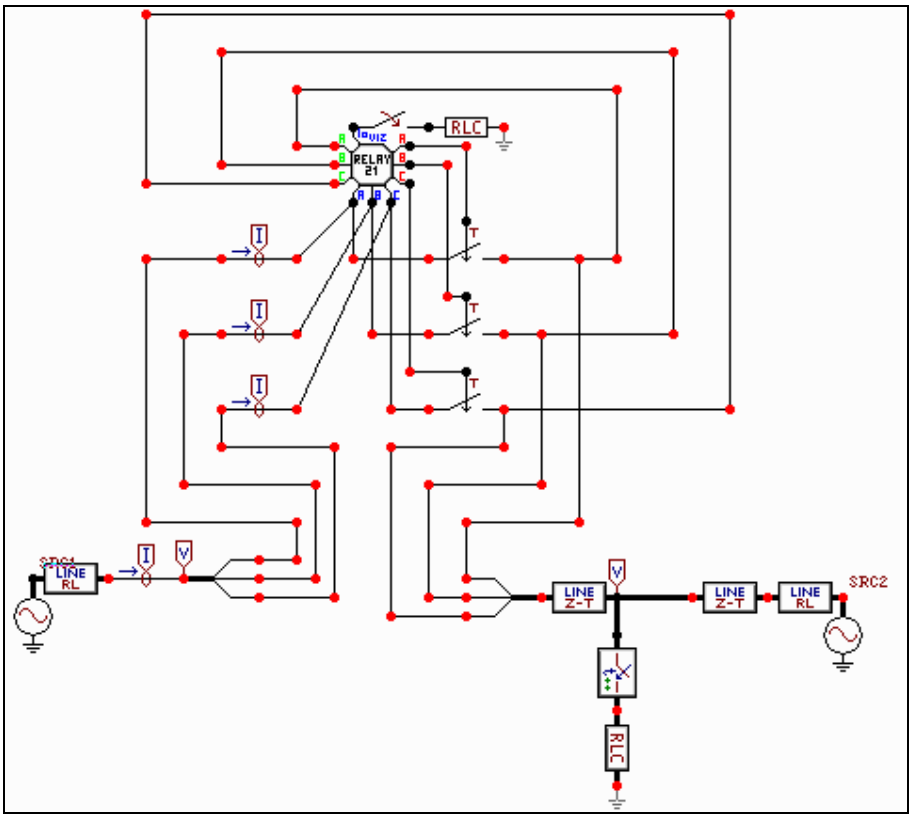

Figura 1: Circuito utilizado para testar a eficácia do elemento *RELAY21*.

5. RESULTADOS DA SIMULAÇÃO

Para testar o funcionamento e aprimorar o algoritmo da função 21 implementada [1], várias ocorrências de falta foram simuladas utilizando-se um sistema com dois geradores e uma linha de 100km no ATP, conforme mostrado na Figura 1.

Neste trabalho é apresentado um caso típico de falta AN, apresentando uma sobrecorrente na fase A, e a resposta do algoritmo da função 21 implementado em linguagem C (Figura 2).

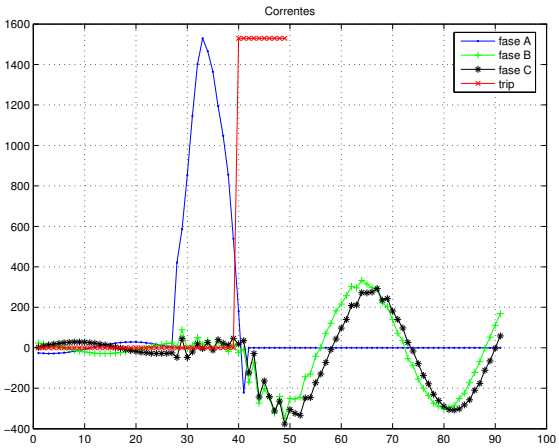

Figura 2: Correntes lidas pelo elemento-relé e sinal de *trip*.

Na Figura 3 a seguir, é possível observar o comportamento das tensões no elemento-relé e o sinal de *trip* dado por este elemento.

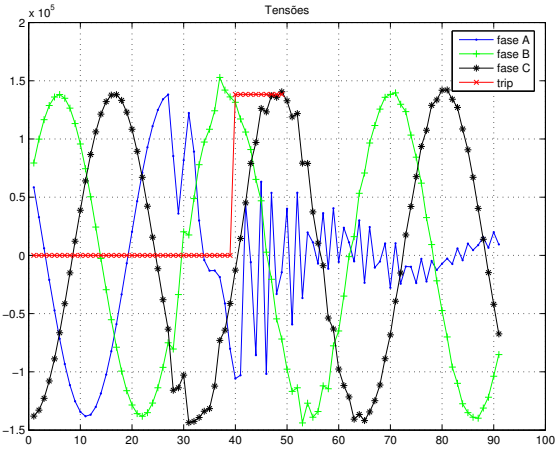

Figura 3: Tensões lidas pelo elemento-relé e sinal de *trip*.

Os resultados apresentados foram colhidos a partir de arquivos gerados pelo próprio programa da função 21 implementado em C. O ATP com sua *MODEL* tradicional permite escrever apenas dois relatórios de dados resultantes em formato ASCII (texto simples) usando os comandos *WRITE1* e *WRITE2*, que criam respectivamente os arquivos *MODELS.1* e *MODELS.2*, enquanto a implementação em C permite infinitas possibilidades de criação de relatórios. No projeto

desenvolvido pela USP, a função 21 enxertada no código do ATP possui um módulo que gera: um relatório mostrando a configuração do relé e seu estado de inicialização; um relatório para análise e depuração de erros do algoritmo; um relatório de dados de entradas e saídas do relé para utilização e análise através do MATLAB e outro relatório de dados de entradas e saídas do relé para ser utilizado pelo Microsoft Excel.

A Figura 4 a seguir mostra a característica MHO e o comportamento das impedâncias de fases durante a simulação. É possível observar que a impedância da fase A mantém-se dentro da característica, sinalizando a existência de um curto-circuito na fase citada.

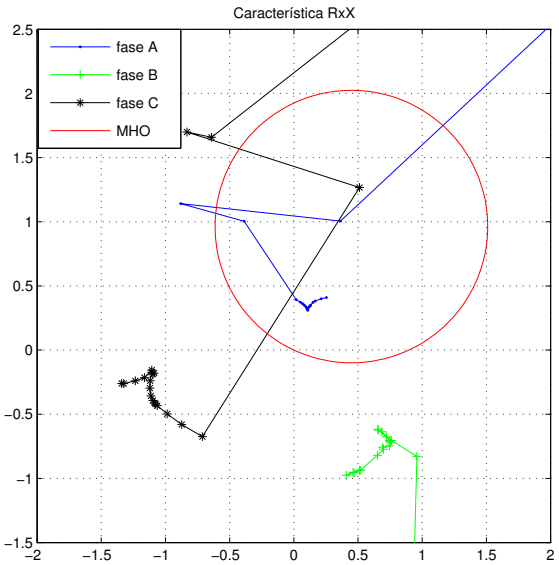

Figura 4: Comportamento das impedâncias de fase perante a característica MHO.

6. ANÁLISE E CONCLUSÃO

Uma das maiores vantagens em se poder codificar em linguagem C dentro do próprio ATP é a interatividade imediata entre o sistema de potência e o sistema de proteção. Além disso, a utilização pelo ATP dos próprios algoritmos a serem implementados no relé, permite a otimização de tempo no desenvolvimento de um protótipo e evita a introdução de erros durante a transcrição do código de MODELS para C ou vice-versa. Antes, havia duas alternativas básicas para se testar um sistema de proteção: ou programava-se o sistema na linguagem da MODELS tradicional, ou simulava-se o sistema em condições pré-determinadas para obter as tensões e correntes resultantes para depois aplicar o algoritmo sobre as formas de onda lidas a partir de um arquivo .LIS. No primeiro caso, havia a necessidade do conhecimento da linguagem da MODELS tradicional e perdia-se algum tempo para implementar em duas linguagens os mesmos algoritmos, uma vez que a programação de *firmware* de relés não é feita em MODULA-2 ou PASCAL, tampouco em FORTRAN. No segundo caso, só se testava até certo ponto a resposta do algoritmo do equipamento a ser desenvolvido mediante tensões e correntes resultantes de uma simulação padronizada e específica, não levando em consideração a interação do equipamento com a situação de falta e pós-falta. Além disso, dependendo das saídas de tensões e correntes do ATP, era necessário desenvolver um programa interpretador de arquivos .LIS para cada caso, o que tornava o trabalho muito dispendioso.

Com a opção de se criar algoritmos em linguagem C e anexá-los a um novo executável do ATP, o trabalho tornou-se mais flexível, simples e rápido, otimizando o tempo de desenvolvimento e implementação de um sistema de proteção. Além disso, o gerenciamento dos resultados torna-se muito mais fácil, pois é possível gerar relatórios de dados em quaisquer formatos, o que facilita a análise e depuração de erros do algoritmo implementado.

Dado que a ferramenta proposta neste trabalho permite simular o algoritmo completo do relé em uma situação de "malha fechada", conclui-se também que ele pode substituir, em parte, os estudos feitos em simuladores em tempo real com custo significativamente mais baixo.

É importante ressaltar novamente que a compilação do núcleo do ATP para a anexação de funções escritas em linguagem C só é disponibilizada para determinadas instituições ou pessoas devidamente licenciadas e mediante consulta aos grupos de usuários do ATP/EMTP.

7. REFERÊNCIAS BIBLIOGRÁFICAS

[1] E.C. Senger et al. "Desenvolvimento de um relé digital de distância", *VII Seminário Técnico de Proteção e Controle*, Rio de Janeiro, Junho de 2003.

[2] G. Furst, "MODELS PRIMER For First Time MODELS Users - Version nº1".

[3] O.P. Hevia, "Compilación del ATP al alcance del usuario." - CAUE-Comité Argentino de Usuarios del EMTP, GISEP-Facultad Regional Santa Fe - UTN Argentina.