INF1413 Teste de Software

Trabalho 4

# Eliminação de defeitos

* 1. **Descrição**: O módulo de XMESSAGE estava reportando a falta de um campo com o índice errado.
  2. **Estratégia**: Ler parte do código que trata os campos em XMESSAGE.CPP
  3. **Diagnose:** O índice do campo começava com 1 em vez de 0. Então se o primeiro campo está faltando em vez de reportar a falha no índice 0 ele reportava no índice 10.
  4. **Resultado:** O teste: == assemble message with no parameter given não falha mais.
  5. **Descrição**: Não era possível inserir um campo tipo string na mensagem.
  6. **Estratégia**: Ler a parte do código que trata com inserção de campos de string.
  7. **Diagnose:** Não observei falhas nos arquivos XMESSAGE.CPP ou XMSGSTR.CPP. Supus que o erro deveria estar no teste em si. Não encontrei erros no script de teste. Olhando o módulo TST\_XMSG, especificadamente a parte que trata do comando =AddStringItem vi que após o tratamento do comando ele insere o campo com um índice errado inxItem + 1 em vez de só inxItem.
  8. **Resultado:** O teste: == insert field and assemble message não falha mais.
  9. **Descrição**: Teste de inserção de vários itens contendo outros campos falhava**.**
  10. **Estratégia**: Começar lendo o script de teste com cuidado.
  11. **Diagnose:** Percebi que no próprio script de teste um dos índices estavam incorretos. =AddStringItem MsgB 6 stringpointerfield " abc"devia ser =AddStringItem MsgB 1 stringpointerfield " abc"
  12. **Resultado:** O teste: == insert field and assemble message não falha mais
  13. **Descrição**: Não era possível substituir uma mensagem**.**
  14. **Estratégia**: Conferir o método ReplaceMessageId no XMESSAGE.CPP.
  15. **Diagnose:** O método ReplaceMessageId não alterava o idMessage, pois estava recriando uma cópia no escopo local.
  16. **Resultado:** O teste: == Replace item and test não falha mais.
  17. **Descrição**: Não era possível inserir um campo com o tempo atual.
  18. **Estratégia**: Conferir os arquivos TST\_XMSG.CPP e XMSGTIME.CPP.
  19. **Diagnose:** Não encontrei nenhum erro no XMSGTIME.CPP. Porém no arquivo TST\_XMSG.CPP na hora de adicionar o tempo atual, o índice do item era incrementado erroneamente
  20. **Resultado:** O teste: == Test message with current time mostra o tempo correto
  21. **Descrição**: O erro que era para ser dado quando muitos campos eram inseridos não estava correto.
  22. **Estratégia**: Olhar as mensagens de erro e onde elas são retornadas.
  23. **Diagnose:** Percebi que havia uma mensagem de erro MSG\_TooManyFields que batia com o resultado esperado do teste. Olhando o método Assemble no XMESSAGE.CPP, percebi que ele estava retornando uma mensagem de erro MSG\_ErrorIllegalIndex quando deveria estar retornando MSG\_TooManyFields.
  24. **Resultado:** O teste: == Test message with recurring field não falha mais.
  25. **Descrição:** O teste de falha por substituição de id de mensagem incorreto falhava
  26. **Estratégia:** Analisar comando de teste =ReplaceMessageId e método ReplaceMessageId no arquivo XMESSAGE.CPP
  27. **Diagnose:** Não foram encontrados defeitos nem no tratamento do comando de teste nem na implementação do ReplaceMessageId. Quando dei uma segunda analisada no script de teste, percebi que estava sendo passada um id de mensagem válida, quando o propósito era realmente testar um id inválido. Troquei o id para um inválido e corrigi a mensagem de erro esperada.
  28. **Resultado:** O teste == Test replace incorrect message id não falha mais.
  29. **Descrição:** O contador de Assemble with error message string – 01 estava zerado.
  30. **Estratégia:** Analisar no código onde esse contador é usado e como escrever um caso de teste que inclui ele.
  31. **Diagnose:** Percebi que esse contador acontecia ao tentar montar uma mensagem com um id de mensagem inválido e um item na última posição no vetor dos itens (20 – 1 = 19). Em seguida escrevi um teste que replica essas condições.
  32. **Resultado:** O contador não está mais zerado.
  33. **Descrição:** O contador de testecontador estava zerado.
  34. **Estratégia:** Investigar o propósito desse contador.
  35. **Diagnose:** Não achei nenhuma referência a ele no código, concluí que deve ter sido colocado por acidente no XMESSAGE.COUNT.
  36. **Resultado:** Todos os testes passam.

# Mutantes

# Os mutantes foram escolhidos em função dos lugares mais prováveis de erro por falta de atenção, nos operadores booleanos onde o controle de fluxo não é tão presente, como em casos de borda (no for) e checando valores de retorno ou nulidade de algum valor.

# Pensamos também em mutar algum delete, mas seria possível que os testes não checassem isso.

## Mutante 1:

# É comum um programador esquecer a semântica dos retornos da biblioteca padrão, então escolhemos mutar a linha 230.

# if ( !mutacao1 ) {

# if ( memcmp( STR\_ID\_NOT\_FOUND , messageString->GetString( ) ,

# strlen( STR\_ID\_NOT\_FOUND )) != 0 )

# {

# DoAssemble( messageString ) ;

# return messageString ;

# } /\* end if \*/

# } /\* end if \*/

# else

# {

# if ( memcmp( STR\_ID\_NOT\_FOUND , messageString->GetString( ) ,

# strlen( STR\_ID\_NOT\_FOUND )) == 0 )

# {

# DoAssemble( messageString ) ;

# return messageString ;

# } /\* end if \*/

# } /\* end else \*/

## Mutante 2:

# Checar se algo é NULL quando deveria não ser é algo comum de se errar por falta de atenção, por isso mutamos a linha 249.

# if ( !mutacao2 ){

# if ( vtItems[ INX\_LAST\_ITEM ] != NULL )

# {

# CNT\_Count( "Assemble with error message string - 01" ) ;

# delete vtItems[ INX\_LAST\_ITEM ] ;

# } /\* end if \*/

# } /\* end if \*/

# else

# {

# if ( vtItems[ INX\_LAST\_ITEM ] == NULL )

# {

# CNT\_Count( "Assemble with error message string - 01" ) ;

# delete vtItems[ INX\_LAST\_ITEM ] ;

# } /\* end if \*/

# } /\* end else \*/

# Mutante 3:

# Casos de borda em for’s normalmente não são checados em testes automatizados e mesmo quando checados os testes podem acusar não testar o caso em que faz diferença o operador de igual estar presente junto ao maior ou menor. Este mutante explora este caso, na linha 267.

# if ( !mutacao3 ) {

# for ( int i = 0 ; i < XMSG\_DIM\_ITEMS - 1 ; i ++ )

# {

# CNT\_Count( "Assemble with error message string - 02" ) ;

# if ( vtItems[ i ] != NULL )

# {

# CNT\_Count( "Assemble with error message string - 03" ) ;

# messageString->Append( Separator ) ;

# sprintf( ASCIINum , "%d" , i ) ;

# messageString->Append( ASCIINum ) ;

# messageString->Append( SeparatorEnd ) ;

# STR\_String \* pItemStr = vtItems[ i ]->ToString() ;

# messageString->Append( pItemStr ) ;

# delete pItemStr ;

# } /\* end if \*/

# } /\* end for \*/

# } /\* end if \*/

# else

# {

# for ( int i = 0 ; i <= XMSG\_DIM\_ITEMS - 1 ; i ++ )

# {

# CNT\_Count( "Assemble with error message string - 02" ) ;

# if ( vtItems[ i ] != NULL )

# {

# CNT\_Count( "Assemble with error message string - 03" ) ;

# messageString->Append( Separator ) ;

# sprintf( ASCIINum , "%d" , i ) ;

# messageString->Append( ASCIINum ) ;

# messageString->Append( SeparatorEnd ) ;

# STR\_String \* pItemStr = vtItems[ i ]->ToString() ;

# messageString->Append( pItemStr ) ;

# delete pItemStr ;

# } /\* end if \*/

# } /\* end for \*/

# } /\* end else \*/

# **Todos os mutantes foram mortos ao fim da execução do script de teste.**