Технология и инструментальные средства испытаний на функциональную безопасность программного обеспечения

Диасамидзе С.В.,

Ковалевский В.В.,

Лозинин А.И.,

Шубинский И.Б.

Требования для оценки программного обеспечения

Требования к документации:

1. Контроль состава и содержания документации:

1.1 Спецификация (ГОСТ 19.202-78)

1.2 Описание программы (ГОСТ 19.402-78)

1.3 Описание применения (ГОСТ 19.502-78)

1.4 Тексты программ, входящих в состав ПО (ГОСТ 19.401-78)

Требования к содержанию испытаний:

2. Контроль исходного состояния ПО

3. Статический анализ исходных текстов программ

3.1 Контроль полноты и отсутствия избыточности исходных текстов

3.2 Контроль соответствия исходных текстов ПО его объектному (загрузочному) коду

3.3 Контроль связей функциональных объектов по управлению

3.4 Контроль связей функциональных объектов по информации

3.5 Контроль информационных объектов

3.6 Контроль наличия заданных конструкций в исходных текстах

3.7 Формирование перечня маршрутов выполнения функциональных объектов

3.8 Анализ критических маршрутов выполнения функциональных объектов

3.9 Анализ алгоритма работы функциональных. объектов на основе блок-схем, построенных по исходным текстам

4. Динамический анализ исходных текстов программ

4.1 Контроль выполнения функциональных объектов

4.2 Сопоставление фактических маршрутов выполнения функциональных объектов и маршрутов, построенных в процессе проведения статического анализа

Основные этапы проведения сертификационных испытаний

Анализ и проверка программной документации

Анализ и проверка требований к ПО в документации требований

Фиксация и установления идентичности текстов программ представленным в программной документации

Разработка и утверждение Программы и методики сертификационных испытаний

Проведение всех необходимых испытаний и тестов ПО с заполнением рабочих протоколов испытаний

Обработка результатов испытаний и заполнение Протокола сертификационных испытаний

Передача документов в орган по сертификации для принятия решения о выдаче сертификата

Цель, стратегия и тактика проведения испытаний программного обеспечения

Целью сертификационных испытаний является контроль отсутствия недекларированных возможностей в программном обеспечении и оценивание соответствия нормативным документам по функциональной безопасности.

Под недекларированными возможностями следует понимать: наличие избыточных функций, которые не декларированы и исполнение которых может привести к тому, что система перестанет должным образом выполнять основные функции (избыточны код, закладки); наличие таких событий, при которых исполнение программы может быть модифицировано, в результате чего система перестанет должным образом выполнять основные функции (несовершенство алгоритма, отсутствие стойкости как к внутренним сбоям и ошибкам, так и к вмешательству извне).

Испытание ПО на отсутствие НДВ – это испытание на отсутствие уязвимостей ПО или, другими словами, расширенные испытания на функциональную безопасность ПО.

Определение, содержит ли данная программа функцию разрушения (нанесения ущерба) - сводится, по сути дела, к задаче исследования программы, задаваемой ее объектным или исполняемым кодом

Под исследованием программы будем понимать именно процесс получения ее алгоритма с последующим сравнением его с декларируемым и проверкой его на уязвимость, стойкость как внешним так и внутренним воздействиям.

При проведения испытаний ПО на основе представленных материалов (исходных текстов и исполняемого ПО) необходимо провести полномасштабную проверку основных функций реализуемых системой и сравнить их набором функций объявленных в спецификации требовании

Все средства исследования ПО можно разбить на два класса: статические и динамические. Первые оперируют исходным кодом программы как данными и строят ее алгоритм без исполнения, вторые же изучают программу, интерпретируя ее в реальной или виртуальной вычислительной среде. Отсюда

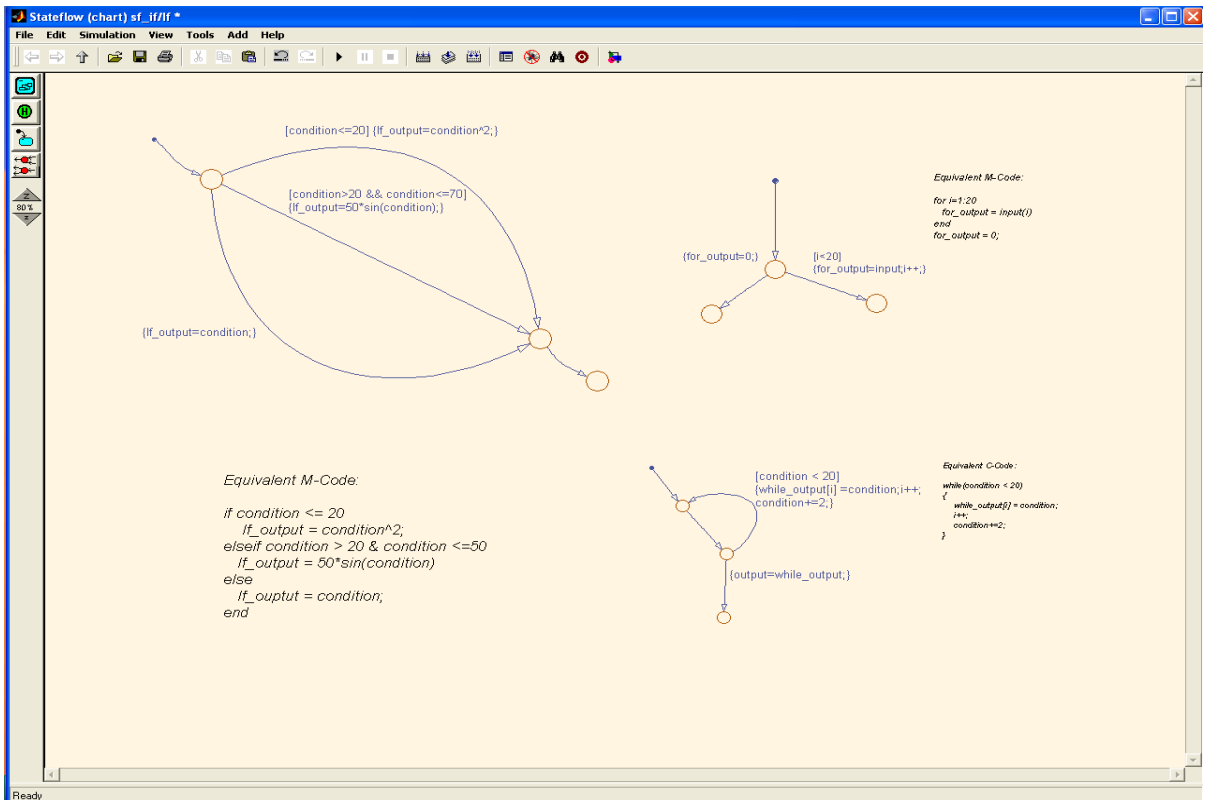
сразу следует, что первые являются более универсальными в том смысле, что теоретически могут получить алгоритм всей программы, в т.ч. и тех блоков, которые никогда не получат управления, вторые же могут строить этот алгоритм только на основании конкретной трассы (trace) программы, полученной при определенных входных данных. Задача получения полного алгоритма программы в этом случае эквивалентна построению исчерпывающего набора тестов для подтверждения правильности программы, что практически невозможно, и вообще при динамическом исследовании можно говорить только о построении некоторой части алгоритма.

Два наиболее известных типа программ, предназначенных для исследования ПО, как раз и относятся к разным классам: это отладчик - динамическое средство и дизассемблер - средство статического исследования. Если первый широко применяется разработчиком для отладки собственных программ и задачи построения алгоритма для него вторичны, то второй формирует на выходе ассемблерный текст алгоритма.

Помимо этих двух основных программных продуктов существуют:

- "дискомпиляторы", генерирующие из исполняемого кода программу на языке высокого уровня;
- "трассировщики", сначала запоминающие каждую инструкцию, проходящую через процессор, а затем переводящие набор их в форму, удобную для статического исследования, автоматически выделяя циклы, подпрограммы и т.п.;
- объединение трассировщика, отладчика и дизассемблера в единое статическо-динамическое средство исследования. Трассировщик в динамике проходит программу до того момента, когда можно применить статическое исследование, попутно запоминая все команды переходов и строя таким образом граф передачи управления. После прохождения по определенному пути графа надо вернуться к ближайшему узлу, сэмулировать ситуацию, при которой возможно прохождение по другой ветви и пойти по ней. Хотя теоретически для построения полного алгоритма здесь тоже необходим полный набор входных данных, их реальное количество будет гораздо меньше, чем для стандартных динамических средств;
- "следящие системы", запоминающие и анализирующие трассу уже не инструкций, а других характеристик, например, вызванных программой прерываний.

Именно такие средства в сочетании с автоматизированной системой семантического анализа предполагается использовать для исследования программ в рамках проведения сертификационных испытаний.



Проведение испытаний на отсутствие НДВ в дополнение с испытаниями на качество ПО микропроцессорных системах существенно повышает возможности устранения уязвимостей в безопасности микропроцессорных систем в целом.

Обзор (перечень) инструментального программного обеспечения, применяемого при проведении сертификационных испытаний ПО.

ПО, используемое при проверке качества алгоритма

MatLab фирмы «MathWorks» - программный пакет с набором расширений предназначенный для имитационного моделирования сложных динамических систем.

В пакете реализован принцип визуально-ориентированного программирования, позволяющий легко создавать и модифицировать модели сложных динамических систем, в том числе содержащие микропроцессорные блоки, при этом сложнейшие уравнения состояния, описывающие работу моделей с или устройств, формируются автоматически.

Пакет позволяет отрабатывать оптимальную стратегию поведения системы в реальном масштабе времени с имитацией тех или иных реальных воздействий окружающей среды, оптимизировать как общую структуру системы, отдельные параметры, так и алгоритм поведения отдельных блоков, в том числе и микропроцессорных блоков, т.е. их программное обеспечение.

На рисунке представлены модели часто употребляемых операторов ветвления if, for и while и эквивалентный им сод на языке MATLAB (Мат, С)

«Modeler 9.0» фирмы OPNET – пакет, позволяющий проводить имитационные исследования цифровых сетей и сетевых приложений.

Применение имитационного моделирования позволяет оценить и оптимизировать характеристики любой конфигурации сети, максимально приближённой к реальной, при любом трафике.

ПО для фиксации и идентификации представленного на сертификацию ПО: Фикс – утилита, позволяющая фиксировать и находить изменение любого бита в документе,, представленном в электронном виде.

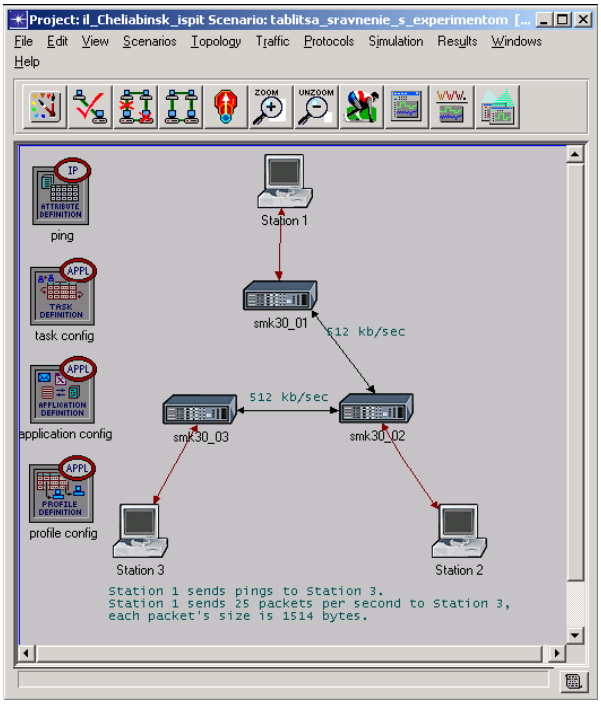
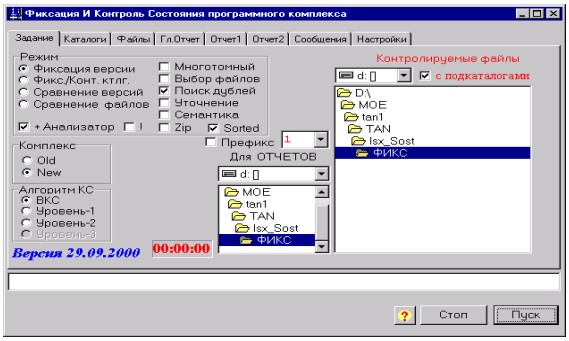


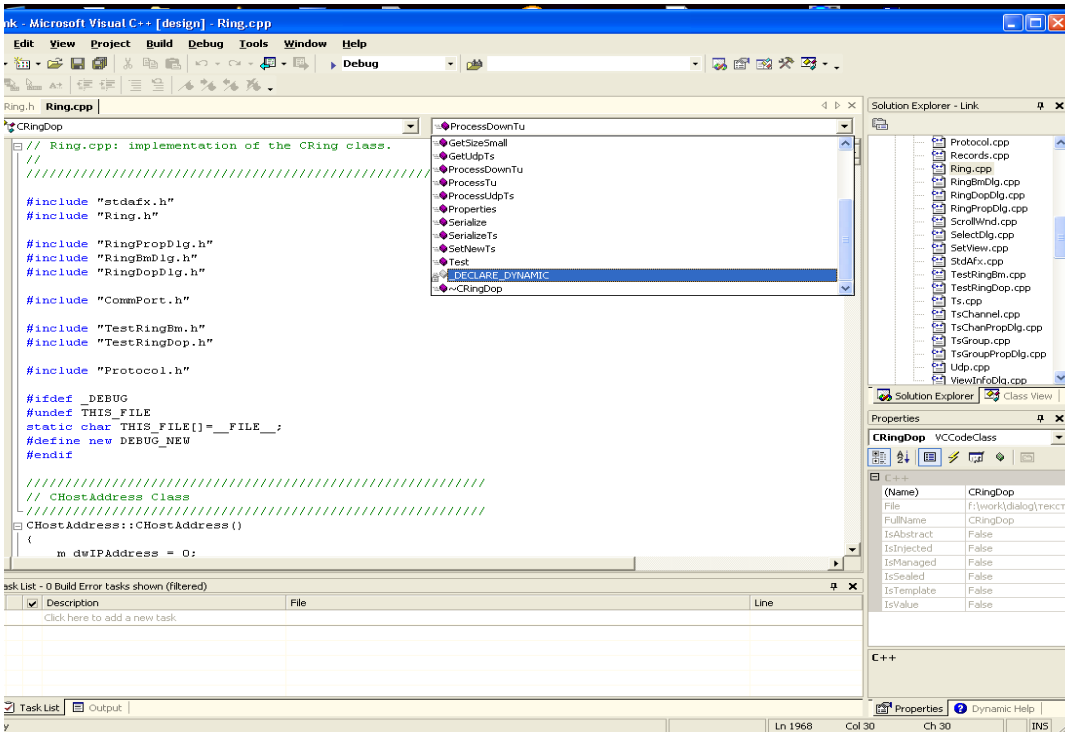
Рис. 2 Моделирование испытательного стенда в программе OPNET Modeler.




```
О Т Ч Е Т о диске N ___
*============================================================*
I Nп/пI ИМЯ файла  I Время записи IДлина,байтIКол.стрI   ВКС  I
I============================================================I
I              Каталог D:\WORK\Is_28147\Rep_1\                I
I------------------------------------------------------------I
I   1 : 2.new         :22.04.98 20-21:     1101 :     - :224e576cI
I   2 : 1.new         :22.04.98 20-21:    38587 :     - :32ef3ceeI
I   3 : report.pas    :22.04.98 20-22:    55089 :   832 :1b4538b7I
I   4 : 4.new         :22.04.98 20-22:    34462 :     - :e14b97f9I
I   5 : _All_Var.rep  :19.04.98 15-04:      274 :     - :67b482e9I
I   6 : _Const.rep    :19.04.98 15-04:      353 :     - :7375c66dI
I   7 : _Var.Rep      :22.04.98 20-22:    47852 :     - :aa0133f9I
I   8 : 2.old         :22.04.98 20-21:     1046 :     - :1050a1bdI
I   9 : 1.old         :22.04.98 20-21:    36318 :     - :c672b2b3I
I  10 : 4.old         :22.04.98 20-21:    32527 :     - :6667865cI
I  Итого: файлов -  10                  247609 :   832 :1024ba29I
I============================================================I
……..
I              Каталог D:\WORK\Is_28147\test\                 I
I------------------------------------------------------------I
I  11 : _DirOld.Dan   :16.04.98 22-59:       28 :     - :1ede9f27I
I  Итого: файлов -   1                       28 :     0 :1ede9f27I
I============================================================I
I  ВСЕГО: файлов -  28                  425425 :  3832 :4ba5ab0bI
```

Среды разработки и сборки прикладного ПО

Development Studio Visual Studio



Среды разработки и сборки ПО для встроенных систем. ПО для проведения статического анализа исходных текстов и исполняемого ПО: Анализаторы исходных текстов, написанных на соответствующем алгоритмическом языке, дисассемблеры, дискомпиляторы.

IDA Pro - интерактивный дизассемблер и отладчик одновременно. Он позволяет превратить бинарный код программы в ассемблерный текст, который может быть применен для анализа работы программы. Хотя IDA и не является дискомпилятором (decompiler), он содержит отладчик (debugger) и может анализировать программы на высоком уровне.

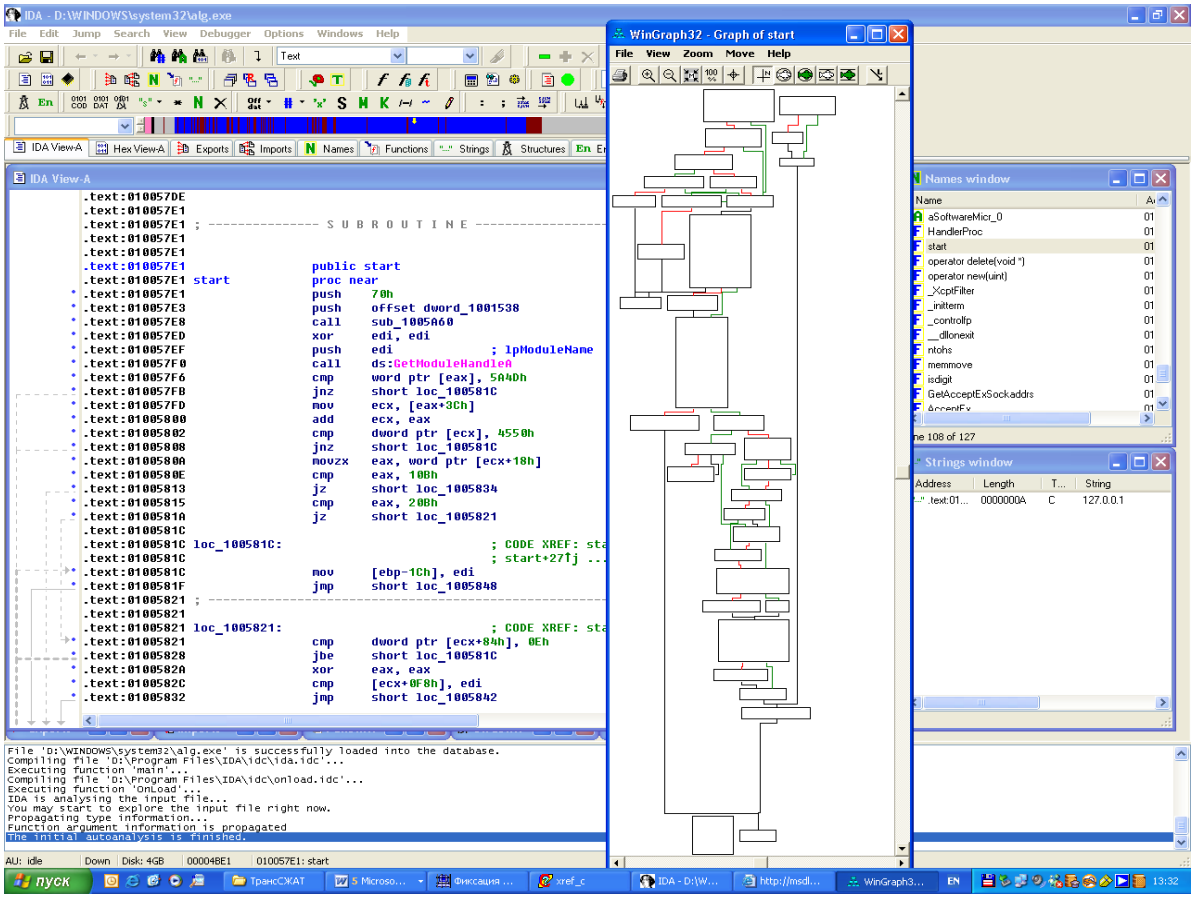
Основная задача - превращение бинарного кода в читаемый текст программы - дополнена многими возможностями, уникальными для этой программы:

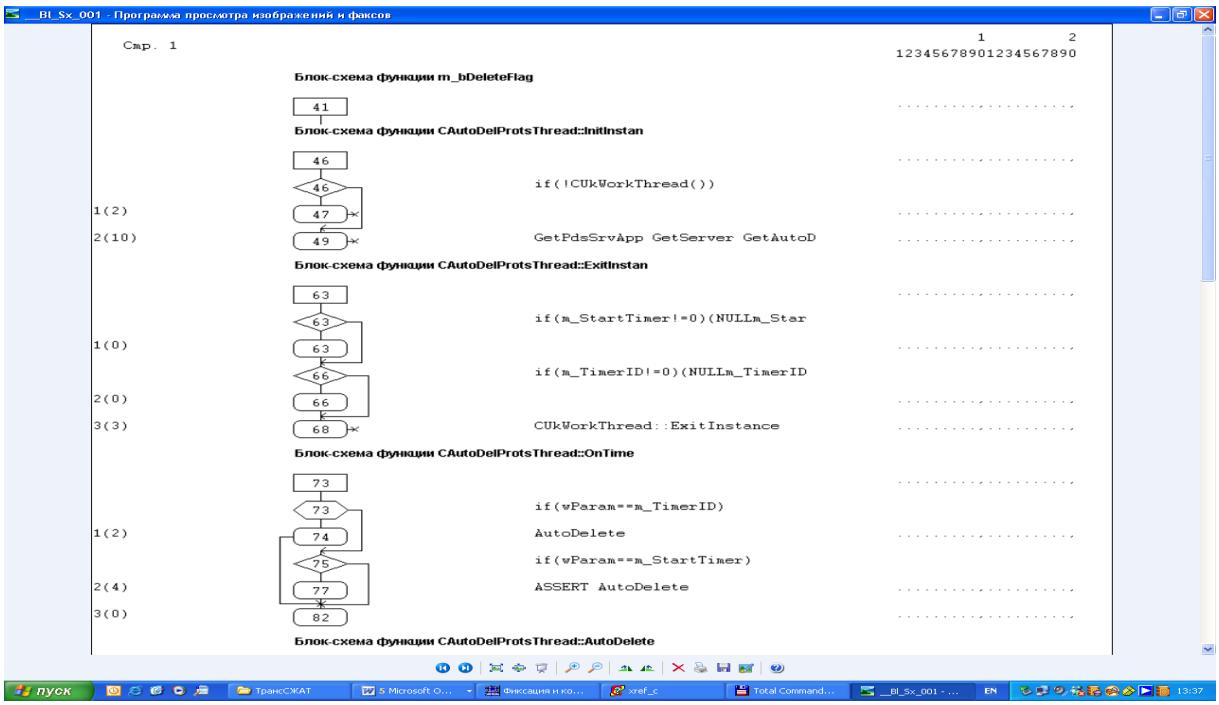
- распознавание стандартных библиотечных функций
- интерактивность работы
- развитая система навигации
- система типов и параметров функций
- встроенный язык программирования IDC
- открытая и модульная архитектура
- возможность работы с практически всеми популярными процессорами
- возможность работы с практически всеми популярными форматами файлов
- работа с структурами данных высокого уровня: массивами, структурами, перечисляемыми типами

- встроенный отладчик для Win32
- Типичные примеры задач, решаемые с помощью дизассемблера:
- анализ вирусов, троянов и других вредоносных программ
- поиск ошибок в программах
- изучение полученного кода
- валидация программ
- оптимизация программ
- разработка защит и поиск дыр в защите

Основные пользователи дизассемблера эксперты по программному обеспечению

Программный комплекс АИСТ-С





Анализаторы различных форматов файлов исполняемого ПО, позволяющие на основе спецификации форматов файлов исполняемого ПО декодировать заголовочную часть исполняемого файла.

ПО для проведения динамического анализа исполняемого ПО

Отладчики (debugger).

Набор тестов, приготовленных с помощью анализатора исходных текстов соответствующего компилятора

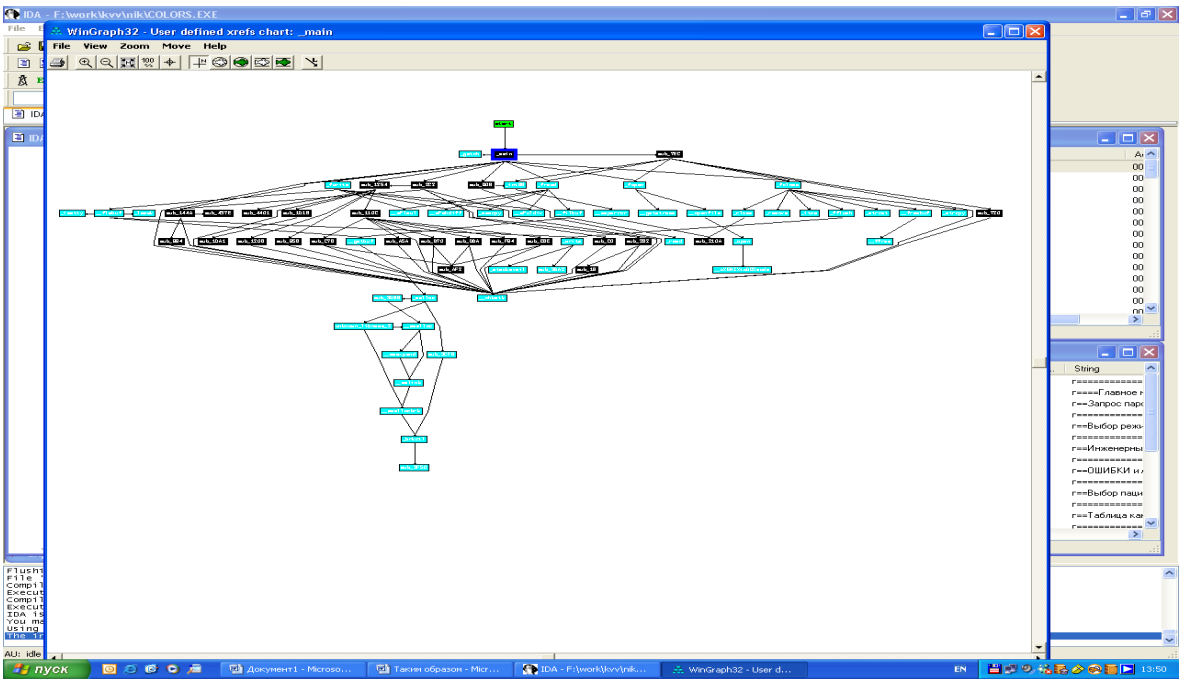
Средства тестирования, в составе IBM Rational Suite – набор средств, предназначенный для анализа работы систем, построенных с помощью языков C/C++, Microsoft Visual Basic, Java, C# .NET, VB. NET и Java .NET:

IBM Rational Robot – средство разработки, записи и выполнения скрипов автоматизированного функционального и регрессионного тестирования приложений, предоставляющее полную поддержку тестирования всех средств управления *Visual Studio.NET*.

IBM Rational XDE Tester — расширенные средства автоматизированного функционального и регрессионного тестирования *Java-* и *Web-приложений* из сред разработки *Eclipse IDE*, *IBM WSAD* и *Rational XDE*.

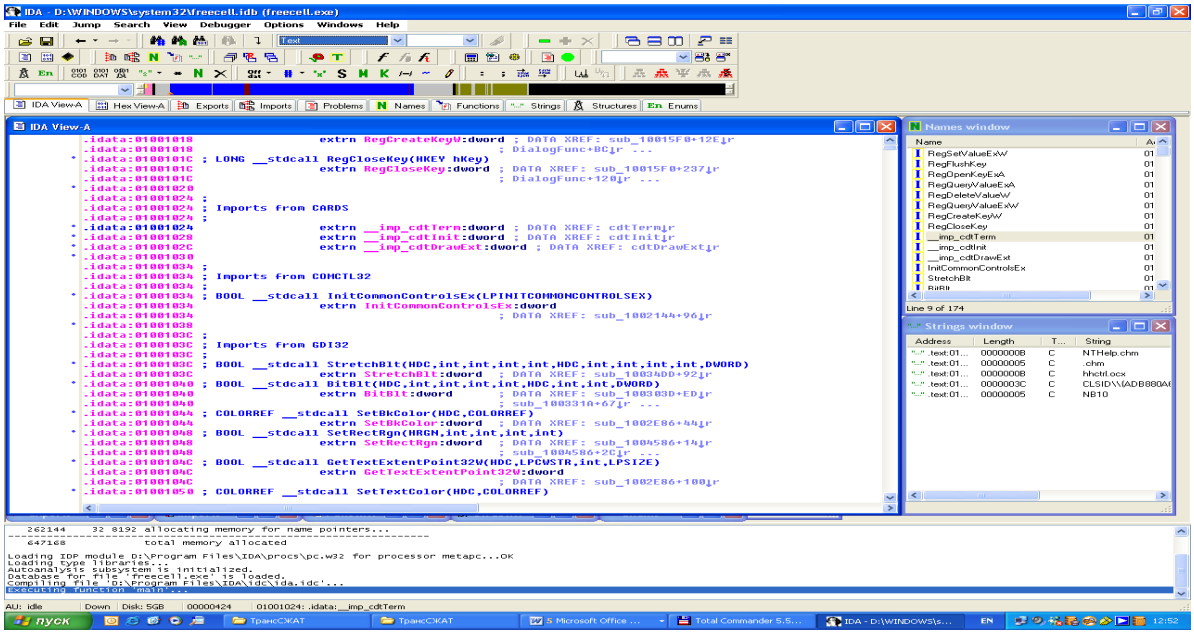
IBM Rational Purify — средство выявления ошибок, связанных с обращением к динамической памяти (версии для Windows и UNIX)

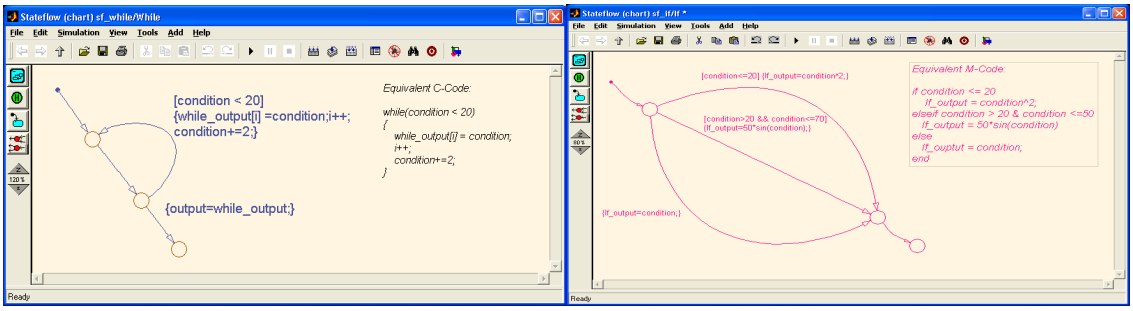
IBM Rational Quantify — средство выявления узких мест в коде, оказывающих влияние на производительность разрабатываемой информационной системы.



IBM Rational PureCoverage — средство определения полноты тестирования кода.

IBM Rational TestFactory — средство для полуавтоматического формирования набора тестовых скриптов, предназначенных для проведения функционального тестирования и обеспечивающих его полноту для конкретной информационной системы; способен выполнить анализ графического интерфейса разрабатываемой ИС и сгенерировать для нее комплексный набор тестов, позволяющий провести максимально полное функциональное тестирование.




Stateflow (chart) sf_while/While
File Edit Simulation View Tools Add Help
[condition < 20]
{while_output[i] =condition;i++;
condition+=2;}
{output=while_output;}
Equivalent C-Code:
while(condition < 20)
{
while_output[i] = condition;
i++;
condition+=2;
}
Ready
Stateflow (chart) sf_if/If *
File Edit Simulation View Tools Add Help
[condition<=20] {If_output=condition^2;}
[condition>20 && condition<=70]
{If_output=50*sin(condition);}
{If_output=condition;}
Equivalent M-Code:
if condition <= 20
If_output = condition^2;
elseif condition > 20 & condition <=50
If_output = 50*sin(condition)
else
If_ouptut = condition;
end
Ready