LDRA
Software Technology

экспонента
ЦЕНТР ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ
И МОДЕЛИРОВАНИЯ

Связь по данным и связь по управлению

Техническая заметка

Автор:
профессор М. А. Хеннелл

ldra.exponenta.ru

Введение

В этом документе описываются функции набора инструментов LDRA, которые вносят вклад в достижение целей **DO-178B / C** «Связь по данным» и «Связь по управлению». В этом разделе приводятся определения и описания этих терминов из стандарта DO-178C, а в следующем разделе описываются некоторые ошибки, которые эти цели предназначены обнаружить. В третьем разделе описывается, как набор инструментов LDRA способствует процессу достижения для этих целей.

- **Связь по данным** — Зависимость компоненты программы от данных, управление которыми осуществляется не только этой компонентой.

- **Связь по управлению** — Вид или степень влияния, оказываемого одной компонентой программы на выполнение другой компоненты.

Кроме того, определение компонента дается как:

- **Компонент** — Автономная часть, комбинация частей, сборочный узел или блок, выполняющая отдельную функцию системы.

В общем, этот термин интерпретируется как включающий: процедуры, функции, подпрограммы, модули и другие подобные конструкции программирования.

Цель, которая должна быть достигнута, таблица A-7, цель 8, описана следующим образом:

- **6.4.4 d. Покрытие структуры программы (связи по данным и по управлению) по результатам испытаний достигается**

Соответствующий раздел стандарта DO-178B (6.4.4.2 Анализ структурного покрытия) описывает следующее:

- **c. В результате данного анализа следует подтвердить связь между компонентами кода по данным и по управлению**

Примеры разъяснений между DO-178B и DO-178C включают:

i. Уточнено, что анализ структурного покрытия связи по данным и управлению между компонентами кода должен быть достигнут путем оценки результатов испытаний на основе требований (см. 6.4.4.2.c).

К сожалению, эти определения не однозначно определяют фактические проверки. Понятно, что они нацелены на ошибки, которые не будут обнаружены другими, более известными методами покрытия.

В следующем обсуждении эти концепции будут подробно изучены, чтобы сформулировать практические методы для достижения этих целей. Затем будет также обсужден вклад, который набор инструментов LDRA может применить к применению этих методов и, следовательно, достижению каждой из целей.

Концепция связи по данным и связи по управлению

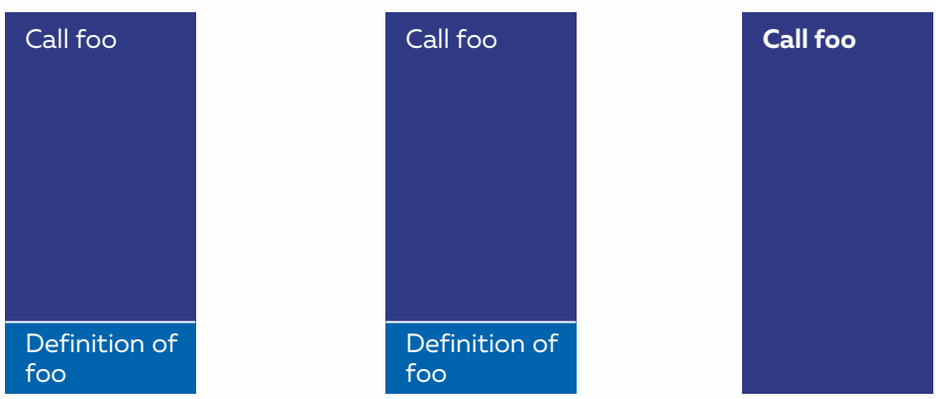
Приведенные выше определения, как правило, недостаточны, чтобы точно определить тип анализа, который требуется. В этом документе используются примеры, иллюстрирующие тип дефекта, который проявляет нежелательное поведение. Затем будет проведено обобщение, чтобы идентифицировать все возможные дефекты этого типа.

Связь по управлению

Во многих отношениях эти проблемы легче всего рассмотреть на примерах:

Пример CC1. Связь по управлению

Рассмотрим следующую модель с тремя файлами, и в каждом из них есть хотя бы один вызов функции foo. Однако, два файла содержат определение функции foo. Эти два определения могут быть идентичными или, возможно, похожими (один и тот же интерфейс).



Компоновщик может решить соотнести все вызовы к определению функции в файле A (вариант 1) или может соотнести вызовы в файле A к определению в файле A, вызов в файле B к определению в файле B, а вызовы в файле C могут быть разрешены случайно (вариант 2). Это пример дефекта связи по управлению, поскольку пользователь может не знать о двусмысленности.

Архитектура системы должна быть в состоянии решить, какой требуется вариант компоновки из этих двух случаев.

Пример CC2

Этот второй пример основан на параметрической связи по данным. Рассмотрим функцию foo, которая имеет процедурный параметр, то есть можно передать имя другой процедуры через список параметров foo.

В этом примере два набора данных, показанные ниже, гарантируют выполнение каждого оператора и дополнительно каждую ветвь (или решение в потоке управления).

var	glob
1	0
0	1

Однако потенциально есть две функции, которые могли быть вызваны в точках, обозначенных как case 1 и case 2, тогда как с этими двумя наборами данных выше в каждой точке вызывается только одна функция, func1 в случае 1 и func2 в случае 2.

```
int var;
void foo ( void (*pfunc)(void) ){
      if( var == 1 ) {
      *pfunc();        /* case 1 */
  } else {
      *pfunc();        /* case 2 */
  }

}
int main ( void ) {
      loop:
        get ( var ); get ( glob );
        if( glob == 1 ){
          foo( &func1 );
        }else{
          foo( &func2 );
        }
      goto loop;
}
```

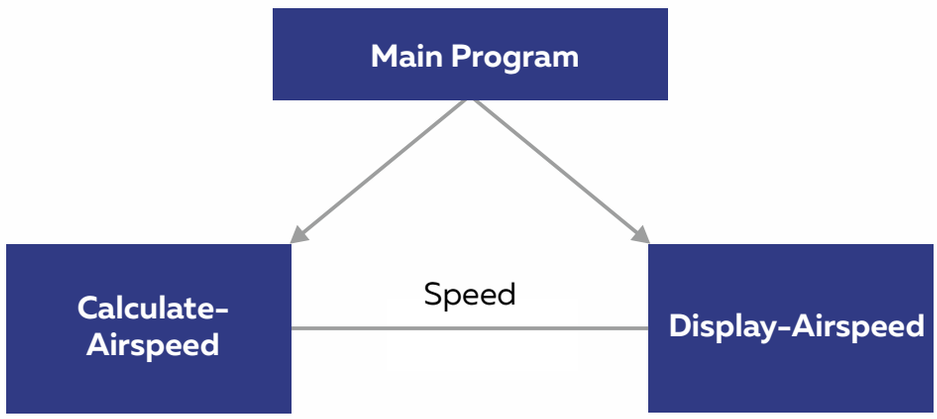
Связь по управлению требует, чтобы все потенциальные вызовы выполнялись в каждой точке. Чтобы гарантировать, что все вызовы потока управления выполняются, тестовые вектора, основанные на требованиях, должны включать два дополнительных набора данных.

Ясно, что эта концепция может применяться ко всем случаям использования указателей на функции. Если функция вызывается разыменованием указателя, все потенциальные функции, которые могут быть вызваны, должны выполняться.

Аналогично в C ++ все виртуальные функции, которые могут быть вызваны в определенной точке, должны выполняться тестовыми векторами, основанными на требованиях.
Чтобы установить связь по управлению, во-первых, важно, чтобы набор возможных функций был вызван, и, во-вторых, должно быть известно, какие члены этого набора вызываются фактически. Эта информация обычно демонстрируется статическим анализом тестируемого кода.

Связь по данным

Связь по данным полностью зависит от структуры диаграммы потока управления как полной системы, так и альтернативных подсистем. Снова рассмотрим некоторые примеры.



Пример DC1. Связь по данным

Для этой демонстрации дефекта связи данных рассмотрим две функции: Calculate-Airspeed и Display-Airspeed, оба из которых вызываются из одной и той же главной программы (диаграмма выше) и которые совместно используют глобальную переменную speed. Calculate-Airspeed вычисляет значение переменной speed, то есть операцию «Set», в то время как функция Display-Airspeed выводит значение speed на устройство отображения, то есть операцию «Use». Основная программа может быть представлена, как показано ниже.

```
int main(){
      int speed, order;
 loop:
      get(order);
      switch( order ){
          case 1: calculate_airspeed(speed);
          case 2: display_airspeed(speed);
      }
      goto loop;
}
```

Возможно создать один тестовый вектор, который выполняет Display-Airspeed, а затем еще один тестовый вектор, который выполняет Calculate-Airspeed. В этом случае связь по управлению проверяется по мере необходимости (т.е. каждый оператор выполняется и каждая ветвь / решение выполняется, в этом случае без требований к MC/DC[1]), но порядок вызовов является дефектным, поскольку Display-Airspeed не имеет действительной скорости для отображения. Точно так же, если тестовый вектор вызывает Calculate-Airspeed, и нет последующего вызова Display-Airspeed, весьма вероятно, что есть еще один дефект.

[1]**MC/DC** — модифицированное покрытие условий решений

```
9Fint
10T foo (
11F int x )
12F  {
13T   if
14T     (
15T      x == 1
16T      )      // a use of x
17T      {
18T       return
19T      x ;    // a use of x
20T     }
21T   else
22T     {
23T       return
24T      x ;    // a use of x
25T     }
26T }
```

Этот пример показывает, что необходимо продемонстрировать, что все входы в данную процедуру имеют допустимые значения в точке вызова и что используются все значения, назначенные глобальным переменным. В этом контексте глобальная переменная является переменной, объявленной и установленной внешне в процедуре, в которой она используется.

Пример DC2.

Рассмотрим следующий код: в следующей функции foo существует три применения параметрической переменной x.

Теперь рассмотрим влияние вызова этой функции из следующей основной программы.

```
28Fint
29T  main()
30F {
31F   int
32F     I ,
33F     y ,
34F     z ;
35T   for
36T     (
37T     i = 1
38T     ;
39T     i <= 2
40T     ;
41T     I ++
```

```
42T     )
43T     {
44T       y = I ;
45F            // y is set
46T    47T       foo (
48T       y ) ;  // y is aliased to x
49T     }
50T   scanf ( "%d" , & z ) ;
51F                 // z is set
52T
53T   foo (
54T   z ) ;  // z is aliased to x
55T }
```

Первый вызов foo с аргументом y приведет к выполнению каждого оператора в foo и дополнительно к каждой ветви/решению, потому что цикл генерирует обе вероятности для y. Это также приведет к выполнению всех трех использований переменной y; Второй вызов foo с аргументом z не охватывает какие-либо утверждения или ветви/решения в foo, которые еще не охвачены, но он вводит еще три использования (из z), из которых один не выполняется. Для тестирования на основе связи по данным потребуются два тестовых вектора: один с z, имеющим единичное значение, и второй случай с другим значением, так что все три использования z в foo (через алиас x) покрываются.

Из этих обсуждений требование DO-178B/C можно формализовать следующим образом:

> Связь по данным включает в себя все глобальные переменные, а также те локальные переменные, которые передаются через списки параметров в компоненты нижнего уровня. Эти последние являются «локально-глобальными» переменными, но впоследствии в этом документе оба этих глобальных типа переменных будут называться просто набором глобальных переменных. Любой из этих глобальных переменных будет присвоено значение некоторой операции, такой как присвоение или получение значения во входной операции. Они называются операциями «Set». Тогда есть точки, в которых эти значения участвуют в операциях (вычисления, выходы и т.д.) Программы, которые называются операциями «Use».

Списки вызовов Set-Call и Call-Use вместе называются «объединения по входу/объединения по выходу переменной» при этом вызове процедуры. Поэтому для каждого вызова процедуры существуют такие объединения, состоящие из всех отдельных списков Set-Call и списков вызовов для всех входных переменных в вызове процедуры.

Если выполняется каждая инструкция в программе, каждая операция Set и каждая операция Use будут выполнены, но это может быть достигнуто путем выполнения только подмножества операций Set для каждого вызова процедуры (см. Предыдущий пример CC2). Существуют два метода улучшения измерения покрытия связи по данным, которое может быть получено из этих списков. Первый - выполнить каждую операцию Set в списках объединения по входу и каждую операцию Use в списках объединения по выходу переменной. Это будет называться покрытием объедениний по входу и выходу переменных. Второй метод состоит в том, чтобы обеспечить выполнение всех комбинаций пар Set-Use в этих списках, то есть покрытие Set-Use. Однако этот метод может привести к "комбинаторному взрыву"[2].

Комитет DO-178B / C уже принял решение против аналогичного комбинаторного взрыва, требуя покрытие MC/DC вместо того, чтобы требовать полного покрытия сочетания условий ветвления (т. е. тестирования всех правильных/ложных комбинаций булевых условий). Поэтому покрытие Set-Use не подходит для практического рассмотрения по тем же причинам практичности. Следствием этого является то, что покрытие связи по данным может быть реализовано практически через покрытие объединений по входу и выходу переменных. Точно так же, как покрытие MC/DC реализует большую часть покрытия условий, так и покрытие объединений по входу и выходу реализует большую часть глобального покрытия Set-Use.

Возвращаясь к случаю глобальных переменных, которые никогда не появляются в списках параметров, видно, что покрытие операторов уже гарантирует, что все записи в их списках Set и Use были выполнены. Однако, это не покрывает все возможные комбинации Set-Use. Поэтому реализация измерения связи по данным, как не комбинаторного метода означает, что необходимо учитывать только те глобальные переменные, которые передаются как параметры.

Реализация и измерение этого показателя очевидна. Рассмотрим трассировку выполнения программного обеспечения во время теста. Для данной переменной будет встречена определенная операция Set, после чего будет встречено использование этой переменной. Это означает, что обе записи в списках объединений по входу и выходу могут быть отмечены как покрытые. Позже может возникнуть одна и та же операция Set, и впоследствии может быть выполнена другая точка использования, и, следовательно, новое использование может быть помечено как покрытое.

Обратите внимание, что это документируется для каждого вызова процедуры, и действительно, каждый вызов процедуры может появляться более одного раза. Это может произойти, если вызов процедуры сам по себе находится в теле другой процедуры, и формальный параметр этой включенной процедуры появляется в списке фактических параметров рассматриваемого вызова функции. Затем эту включенную процедуру необходимо оценить для всех соответствующих фактических параметров внешнего вызова.

Дополнительные затраты, связанные с измерением связи по данным, сложно оценить из-за влияния покрытия операторов/решений, покрытия вызовов процедур и покрытия MC/DC, которые неизбежно будут порождать больше записей в списках объединений по входам и выходам переменных, чем при просто переменных. На практике может быть разумной стратегией исследовать связь по данным только после того, как эти показатели покрытия были максимально увеличены до требуемых уровней.

Цель DO-178C состоит в том, чтобы все связи по данным удовлетворялись данными испытаний, основанных на требованиях, и что не должно быть операций Use, которым не предшествует операция Set.

[2]**Комбинаторный взрыв** — термин, используемый для описания эффекта резкого («взрывного») роста временной сложности алгоритма при увеличении размера входных данных задачи

Набор инструментов LDRA

В этом разделе описываются функции набора инструментов LDRA, которые вносят вклад в достижение целей DO-178B/C «Связь по данным» и «Связь по управлению». Есть два вопроса, которые необходимо учитывать. Во-первых, как определить потребность в специальных тестах, а во-вторых, как можно показать конкретные тестовые вектора для обеспечения необходимого покрытия? Набор инструментов LDRA решает первую проблему с помощью Статического анализа, а во втором случае объединяет знания, полученные в результате этого Статического анализа, с детальным динамическим анализом потока управления.

Связь по управлению

Ключевым элементом средств анализа структурного покрытия, предоставляемых набором инструментов LDRA, является модель потока управления, которая автоматически определяется посредством обширного статического анализа тестируемого исходного кода. При составлении отчета об анализе структурного покрытия набор инструментов затем сравнивает фактический поток управления с прогнозируемым статическим анализом. Это позволяет инструменту идентифицировать любые ошибки в компоновке или других отклонениях от требуемого поведения потока управления. Ошибки компоновки могут возникать, например, если одна и та же процедура или имя функции встречаются в нескольких файлах исходного кода. Именно от компоновщика зависит, какие процедуры или функции будут соотнесены с конкретными вызовами. Набор инструментов LDRA предоставляет предупреждения на основе статического анализа о возможной двусмысленности, а затем создает генерируемые динамическим анализом сообщения на основе тестирования, основанного на требованиях, если фактическая связь отличается от прогнозируемой. Затем пользователь исследует графические представления, чтобы решить, что предсказывали проектные или архитектурные представления.

В дополнении к этому основному аспекту связи по управлению набор инструментов LDRA также собирает покрытие вызовов процедур и предоставляет графическую индикацию связей по управлению при помощи графика «Callgraph» (Callgraph Display). Это средство обеспечивает визуальное представление зависимости данного программного компонента от тех компонентов, которые его вызывают. Затем пользователи Callgraph Display могут сфокусироваться на отдельных связях по управлению, выбирая индивидуальный программный компонент (процедурный узел) и выбирая опцию «Nuclear Spider» в меню правой кнопкой мыши. Этот график обеспечивает графическое представление непосредственной связи по управлению, а расширенная или иерархическая связь может затем отображаться при помощи «Extended Spider», который выбирается из того же меню. В качестве альтернативы, вызывающие компоненты вместе с частотой вызова могут быть перечислены в текстовом формате с помощью опции «Interface Information», которая снова доступна из того же меню.

Эта информация также может быть отображена непосредственно в исходном коде путем «развертывания» до конкретных предикатов в исходном коде, которые должны быть выполнены, чтобы повлиять на вызов. Это может быть достигнуто либо непосредственно из Callgraph Display, либо через промежуточный шаг выбора «Processural Flowgraph Display», а затем щелчком по соответствующему узлу графа.

Следует также отметить, что точный семантический анализ может использоваться для дополнительного мониторинга глобальных переменных, если это будет считаться желательным. Например, могут быть добавлены аннотации, которые будут проверять фактические значения указателя на функцию.

Для приведенных ранее примеров:

Пример CC1

Набор инструментов LDRA сообщает о неоднозначности, вызванной двумя определениями, и наглядно показывает, что связь, которую он предполагает, является правильной. Затем пользователь может либо подтвердить, либо опровергнуть эту модель привязки и устранить неоднозначность. В качестве альтернативы, динамический анализ покажет фактическую связь с точки зрения покрытия. Для case 1 определение процедуры в файле B будет иметь нулевое покрытие операторов и ветвей, и все покрытие будет отображаться в определении в файле A. В случае 2 оба определения будут демонстрировать покрытие. Какое разрешение используется вызовами в файле C, будет показано в таблице покрытия вызовов процедур? Поэтому дефект может быть обнаружен.

Пример CC2

В этом примере набор инструментов LDRA статически сканирует все вызовы процедуры foo и получает все функции, которые передаются как фактические параметры. В последующем анализе потоков данных и управления будет использоваться список вызываемых функций. Этот анализ будет отслеживать вызываемые функции по всему графу, используя другие вызовы процедур, если это необходимо.

В заключение, верификация связей по управлению значительно упрощена благодаря обширным проверкам статического анализа, которые затем повышают доверие к покрытию связей по управлению, полученному в результате динамического анализа покрытия тестовыми данными на основе требований.

Связь по данным

Набор инструментов LDRA устраняет дефекты связи по данным двумя способами: статическим анализом и динамическим анализом. Сила этих двух методов сочетается, что дает соответствующие показатели покрытия.

Во-первых, набор инструментов применяет общесистемный статистический анализ потока данных к полному графическому представлению системного теста и идентифицирует не только упомянутые ранее точки Set and Use, но также пары Dec-Use (объявление для использования путей), пути Set-Set и пути Set-Unuse (назначение пути переменной вне области видимости). Обратите внимание, что точки использования иногда упоминаются как точки «де-определения» (Undefine) переменных. Из всех пар, пары Dec-Use и Set-Unuse являются четкими индикаторами дефектов. Пары Set-Set являются частыми, но иногда они могут указывать на дефект, и, в общем, их тяжело удалить, поскольку они подвержены комбинаторике. Обратите внимание, что в обозначениях традиционного анализа потока данных пары Dec-Use сопоставляются с аномалиями UR, пары Set-Set - DD аномалиями, а пары Set-Unuse – DU- аномалиями.

Поскольку присвоение значений переменным часто тесно связано с операциями ввода-вывода, крайне важно не иметь дефектов в операциях ввода-вывода. Поэтому набор инструментов также выполняет поиск по всему общему графу потока управления, чтобы показать, что все операции ввода-вывода являются подходящими, связывая их с шаблонами использования входных переменных. Проверки операций с файлами гарантируют, что в каждом пути исполнения всем чтениям и записи предшествует открытие файла, а уже открытый файл не открывается повторно и все открытые файлы закрываются при завершении программы. Также будут проверены запись в ранее закрытый файл или закрытие ранее закрытого файла и другие дефекты ввода-вывода.

Возможно, что анализ создает ненужные пары (из всех обсуждаемых типов) из-за наличия неосуществимых путей (называемых недостижимыми путями в DO-178B/C). Набор инструментов сообщает о многих причинах неосуществимых путей в статическом анализе. Это позволяет разработчикам программного обеспечения упростить свой код, что, в свою очередь, помогает снизить затраты на построение тестовых векторов. Попытки выполнить компоненты кода, чтобы получить покрытие определенной пары Set-Use только для того, чтобы обнаружить, что пути, которые их соединяют являются неосуществимыми, может быть разочаровывающими.

Анализ динамического потока данных затем идентифицирует фактические (как видно из потока управления) аномалии Dec-Use и фактические аномалии Set-Unuse, поскольку они могут указывать на возможное неправильное упорядочение компонентов. Дефект Dec-Use (т.е. неинициализированная переменная) всегда является ошибкой, тогда как дефект Set-Unuse может возникать из-за неуклюжего стиля программирования. Эта динамическая информация получается из трассировки пути выполнения, так как программа выполняет тестовые данные и, следовательно, обеспечивает независимое подтверждение результатов, представленных в статическом анализе потока данных. Обратите внимание, что, как правило, более экономически оправдано устранение этих дефектов после проведения статического анализа и до начала тестирования. Списки Set-Use and Use-Call и их покрытие документируются, как показано на диаграмме 1.

```
=============================================================================================
VARIABLE       DPTH PARAM    FILE          PROCEDURE     TYPE   ATTRIBUTE
NAME                ALIAS                NAME          CODE   CODE       USED ON LINES..
=============================================================================================
i                               ex_dc2.c    main          L      E          32
                                                          L      R          39     41
                                                                            44
                                                          L      D          37     41

--------
x                               ex_dc2.c    foo           P      E          11
                                                          P      R          15     19
                                                                            24

--------
y                               ex_dc2.c    main          L      E          33
                                                          L      R          48
                                                          L      D          44
                     1  x        ex_dc2.c    foo           P      E          11
                                                          P      R          15     19
                                                                            24

--------
z                               ex_dc2.c    main          L      E          34
                                                          L      I          50
                                                          L      R          54
                                                          L      D          50
                     1  x        ex_dc2.c    foo           P      E          11
                                                          P      R          15
                                                                            19  *****
                                                                            24
```

Диаграмма 1

Эти результаты получены на основе динамического анализа фактического пути выполнения, полученного из последнего запуска программы. Номера строк, за которыми следует ***, не выполнялись на любом испытательном прогоне до настоящего времени.**

Разъяснение терминов, используемых на **Диаграмме 1**, можно увидеть на **Диаграмме 2.**

```
                                    ******************
                                    *                  *
                                    *  Key to Terms    *
                                    *                  *
                                    ******************

Type and Attribute codes are shown for each variable
----------------------------------------------------

Variable Type Codes
-------------------
     ======================
     TYPE           MEANING
     ======================
     C              Constant
     L              Local
     G              Global
     P              Parameter
     LG             Local-Global

Variable Attribute Codes
------------------------
     ======================
     ATTRIBUTE      MEANING
     ======================
     E              Declaration
     D              Definition
     R              Reference
     I              Input
     O              Output
     N              Indirect Usage
     U              Unused parameter


Other Terms in Dynamic Data Flow Table
--------------------------------------
======================
ITEM        DESCRIPTION
======================
Variable    Main variable name matching user selection
Alias       Call depth + aliased name when passed as procedure parameter
n           Variable used on line n
n *****     Use of variable on line n not hit in any data set
            Use of aliased parameter on line n not hit for function call
(n)         Use of variable on line n hit in last data set
n -         Variable used on non-executable line n
```

Диаграмма 2

Набор инструментов LDRA предоставляет подробную информацию о связи по данным, как в статическом, так и в динамическом доменах, как описано ниже, чтобы помочь разработчикам выбрать тестовые данные, основанные на требованиях, для выполнения любых непокрытых элементов в списках объединений по входу/выходу.

Некоторые аспекты связи по данным снова могут быть доступны в графах Callgraph Display и Flowgraph Display, где пользователи могут «развернуть» исходный код, содержащий конкретные экземпляры предикатов внутри кода, которые влияют на поведение зависимого программного компонента.

Анализ перекрестных ссылок и анализ объектов данных может использоваться для отображения ВСЕХ экземпляров элементов данных, к которым обращается конкретный программный компонент. Это включает в себя локальные переменные, объявленные в пределах компонента и глобальные переменные, к которым обращается компонент, но объявленные в другом месте. Примечательно, что с помощью сложного модуля анализа объектов данных набор инструментов LDRA способен обеспечить «отфильтрованный» и, следовательно, чрезвычайно сосредоточенный анализ, который будет отслеживать и сообщать пользовательские элементы данных в границах файлов и процедур даже в тех случаях, когда они передаются через алиасы как параметры для вызова процедур или к ним осуществляется доступ через цепочку вызовов. С добавлением вывода анализа потока информации этот механизм отчетности дополняется, включая как прямую, так и условную зависимость от объекта данных и таблицы обратной зависимости.

Помимо внесения вклада в аспекты отчетов анализа данных, анализ потока информации будет определять и сообщать о зависимостях данных в анализируемом коде, а также автоматически обнаруживать и сообщать о ненадлежащих (неожиданных) зависимостях данных. Затем, возможность определять и проверять интерфейсы данных между модулями и компонентами обеспечивается модулем статического анализа потока данных, который документирует интерфейс каждой процедуры в исходном коде с точки зрения его параметров, глобальных переменных и любых возвращенных результатов. Каждый из этих элементов указан вместе с его определенным использованием, т. е. был ли только обращение к объекту, или он был изменен в процедуре или он не используется. В динамическом домене средство Dynamic Data Flow Coverage, предоставляемое набором инструментов LDRA, указывает, какие компоненты данных были доступны во время выполнения тестов. При этом он использует трассировку выполнения, связанную с каждым конкретным набором тестовых данных, и тем самым обеспечивает соединение данных для этого конкретного тестового примера, а также накопленные результаты для всех прогонов тестов.

Анализ связи по данным - метод интеграции программного обеспечения, поскольку он рассматривает способ передачи данных через систему. Существует три способа интеграции:

- **Интеграция всей системы за одну операцию**
- **Интеграция снизу вверх, тестируя каждый уровень с использованием драйверов**
- **Интеграция сверху вниз, тестируя каждый уровень с помощью заглушек.**

Набор инструментов LDRA может выполнить анализ данных при любой из этих трех альтернатив. Он может автоматически генерировать как драйверы, так и «заглушки» с минимальным вовлечением пользователя, что существенно снижает временные затраты.

Покрытие динамического потока данных также может отображать каждый экземпляр инициализации или операции Set для глобальной переменной и всех последующих применений этой переменной. Это списки объединений по входу и выходу переменных. Они включают в себя все экранирования глобальной переменной, поскольку она затем передается через интерфейсы процедур в списках параметров. Полный набор объединений по входу и выходу для каждой глобальной переменной и локальных переменных, передаваемых через списки параметров, генерируется для любого из трех перечисленных выше параметров интеграции. Затем, из анализа динамического покрытия, который генерируется с использованием тестовых векторов, набор инструментов сообщает, какие экземпляры Set или Use были покрыты тестами и, что наиболее критично, которые не были покрыты.

Из результирующего покрытия можно обнаружить, когда все экземпляры объединений по входу и выходу переменной были выполнены, и, следовательно, удовлетворена цель DO-178C.

В заключение набор инструментов LDRA предоставляет исключительно подробные проверки на основе статического анализа, чтобы уменьшить вероятность дефектов, которые затрудняют задачу оценки связи по данным. Динамический анализ обеспечивает явное покрытие связей для тестовых данных.

Вывод

В этом документе описываются обширные функции набора инструментов LDRA, которые могут подтвердить покрытие связей по данным и управлению.

Он показывает, что набор инструментов LDRA может предоставить информацию и анализ для достижения целей DO-178C по связям по данным и управлению.

Приложение А

Для иллюстративного руководства по подробным проверкам необходима следующая таблица из «Developing Safety-critical Software» Leanna Rierson, CRC Press, 2013, ISBN 978-1-4398-1368-3. В соответствующих разделах ниже будет показан вклад набора инструментов LDRA в каждый отмеченный элемент.

Примеры элементов связи по данным для рассмотрения	Примеры элементов связи
1. Все внешние входы и выходы определены и правильны	1. Порядок выполнения
2. Все внутренние входы и выходы определены и правильны	2. Скорость выполнения
3. Типы данных правильны и непротиворечивы	3. Условное исполнение идентифицировано и правильно
4. Модули согласованы и согласовываются со словарем данных	4. Внешние зависимости установлены и правильны
5. Словарь данных и код согласованы и оба завершены	5. Порядок,скорость и условия выполнения удовлетворяют требованиям
6. Данные отправляются и принимаются в правильном порядке	6. Прерывания идентифицированы и правильны
7. Данные используются непротиворечиво	7. Исключения идентифицированы и правильны
8. Повреждение данных предотвращено или обнаружено	8. Сбросы идентифицированы и правильны
9. Данные инициализируются или считываются перед использованием	9. Реакции на прерывания питания идентифицированы и правильны
10. Предупреждено или обнаружено использование устаревших или недействительных данных	10. Планировщики переднего плана выполняются в правильном порядке и по правильной ставке
11. Предотвращены или обнаружены ошибки данных	11. Фоновые планировщики выполняются и не входят в бесконечный цикл
12. Неожиданные значения с плавающей запятой предотвращаются или обнаруживаются	12. Код соответствует архитектуре
13. Параметры передаются правильно	
14. Глобальные данные и элементы данных в глобальных конструкциях данных верны	
15. Доступ к вводу / выводу из внешних источников осуществляется правильно	
16. Все переменные устанавливаются (или инициализируются) перед использованием	
17. Используются все переменные	
18. Переполнения обнаружены и исправлены	
19. Используются локальные и глобальные данные	
20. Массивы индексируются правильно	
21. Код соответствует архитектуре	

Элементы в таблице 9 выше могут быть рассмотрены следующим образом:

1. Все внешние входы и выходы определены и правильны. Анализ интерфейса процедуры показывает детали параметров каждого компонента, возвращаемых значений, операций ввода-вывода и глобальных переменных. Эта информация чрезвычайно полезна для анализа и рассмотрения архитектуры.

2. Все внутренние входы и выходы определены и правильны. Анализ интерфейса процедуры показывает детали всех переменных, сопоставленных с оборудованием и переменными, которые отображаются в списках параметров для подкомпонентов или операторов вывода.

3. Типы данных правильны и непротиворечивы. Проверка сильной типизации - это мощная функция инструментов, вызывающих до 340 проверок объявлений и 166 проверок выражений.

4. Модули согласованы и согласовываются со словарем данных Набор инструментов LDRA проверяет, что интерфейсы между блоками удовлетворяют приблизительно 90 проверкам количества, типа и атрибутов объектов в интерфейсах.

5. Словарь данных и код согласны и оба завершены. Инструмент обеспечивает соответствие всех требований коду и коду ко всем требованиям, которые являются формой полноты.

6. Данные отправляются и принимаются в правильном порядке. Существуют проверки того, что чтение, запись и закрытие файлов всегда осуществляются в правильном порядке. Другие проверки гарантируют, что связанные компоненты выполняются во всех путях исполнения в предопределенном порядке. Существует множество проверок для выявления таких ошибок порядка, поскольку переменные являются переназначенными значениями без промежуточного использования.

7. Данные используются непротиворечиво. Есть 166 проверок структуры выражений.

8. Повреждение данных предотвращено и обнаружено. Существуют проверки переполнения массива и использование перекрывающихся объявлений.

9. Данные инициализируются или считываются перед использованием. Системный анализ потока данных в наборе инструментов LDRA обеспечивает формальное доказательство того, что это ограничение выполнено.

10. Предупреждено или обнаружено использование устаревших или недействительных данных. Набор инструментов LDRA проверяет, что все входные данные проверяются перед использованием (т.е. подвержены ограничениям).

11. Предотвращены или обнаружены ошибки данных.

12. Неожиданные значения с плавающей запятой предотвращаются или обнаруживаются.

13. Параметры передаются правильно. Выполняется около 46 проверок параметров.

14. Глобальные данные и элементы данных в глобальных конструкциях данных верны. Сильная типизация также выполняется над полями структур данных, гарантируя их правильную инициализацию и использование.

15. Доступ к вводу / выводу из внешних источников осуществляется правильно. Проверки выполняются над порядком открытия, чтения, записи и закрытия файла. Входные значения из чтения файла проверяются, чтобы убедиться, что их значения были санированы на всех путях перед использованием.

16. Все переменные устанавливаются (или инициализируются) перед использованием. Системный анализ потока данных в наборе инструментов LDRA обеспечивает формальное доказательство того, что это ограничение выполнено.

17. Используются все переменные. Проверки гарантируют, что всем переменным, которым задано значение, используются до выхода из области видимости.

18. Переполнения обнаружены и исправлены. Набор инструментов LDRA сообщает о некоторых случаях потенциального переполнения целого числа.

19. Используются локальные и глобальные данные. Проверяется, что используются все именованные местоположения, содержащие данные.

20. Массивы индексируются правильно. Проверки границ массивов выполняются статически. Существует также возможность динамической проверки.

21. Код соответствует архитектуре. Графические средства помогают сопоставить код с представлением архитектуры.

В дополнение к средствам статического анализа, описанным выше, в динамическом домене, инструмент Dynamic Callgraph Display включает в себя информацию анализа структурного покрытия, чтобы продемонстрировать степень, в которой идентифицированная связь по управлению была идентифицирована во время исполнения.

Источники информации, описанные выше, могут быть использованы для идентификации зависимостей управления. Дополнительные, не графические, данные, такие как данные, сгенерированные модулем анализа потока информации LDRA и модулем анализа объектов данных, позволяют дополнительно улучшить доступные средства отчетности, идентифицируя зависимости объектов данных модулей или компонентов.

1. Порядок выполнения идентифицирован и правилен. 85 проверок.
2. Скорость выполнения идентифицирована и правильна.
3. Условное исполнение идентифицировано и правильно. 27 проверок.
4. Внешние зависимости установлены и правильны.
5. Порядок, скорость и условия выполнения удовлетворяют требованиям.
6. Прерывания обнаружены и правильны 17 проверок.
7. Исключения идентифицированы и правильны.
8. Сбросы идентифицированы и правильны.
9. Реакции на прерывания питания идентифицированы и правильны.
10. Планировщики переднего плана выполняются в правильном порядке и по правильной ставке.
11. Фоновые планировщики выполняются и не входят в бесконечный цикл.
    Бесконечные циклы обнаружены. 26 проверок.
12. Код соответствует архитектуре.

LDRA Software Technology

экспонента
ЦЕНТР ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ
И МОДЕЛИРОВАНИЯ

+7 (495) 009-65-85

info@exponenta.ru

ldra.exponenta.ru