РАЗРАБОТКА ИНСТРУМЕНТАРИЯ СБОРА И АНАЛИЗА ПОКРЫТИЯ БОРТОВОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Козырев В.П., Петров М.Е., Судьбин М.А. (Advalange),
Солоделов Ю.А., Соловьева Е.А. (ГосНИИАС)

Разработка ОСРВ JetOS

Инструментарий сбора и анализа покрытия бортового программного обеспечения разрабатывается в рамках НИР ГосНИИАС по созданию операционной системы реального времени **JetOS**.

Важнейшее требование к ОСРВ (как и к любому авиационному бортовому ПО) – **сертифицируемость** в соответствии с КТ-178С

- Процессы разработки
- Требования высокого уровня
- Проект ПО
- Тестирование
- **Анализ структурного покрытия**
- Анализ характеристик
- Гарантия качества
- Управление конфигурацией артефактов жизненного цикла
- <…>

Требования к ОСРВ JetOS

Помимо сертифицируемости, к JetOS предъявляется ряд требований:

- Поддержка API ARINC 653 для функциональных приложений:
  - Статическая конфигурация – по доступу к памяти, по обмену между приложениями и т.д.
  - Гарантия разделения приложений по ресурсам (в том числе по времени и памяти)
- Поддержка многоядерности
- Легкая переносимость между аппаратными платформами
- Наличие «рабочего места разработчика ФПО» с необходимыми при разработке инструментами

Цель НИР по созданию ОСРВ JetOS

Основная цель – получение платформонезависимой части ОСРВ с набором сертификационных данных (**сертификационным пакетом**):

- Ядро *(kernel)*;
- Пользовательская библиотека *(libjet)*.

Сертификационный пакет включает в себя (помимо прочего) планы и стандарты, требования высокого уровня, Проект ПО, протоколы инспекций, записи гарантии качества, протоколы верификации и т.д. и т.п.

В дальнейшем сертификационный пакет будет включен в состав сертификационного пакета комплектующего изделия (бортовой системы).

Участники НИР по созданию ОСРВ JetOS

НИР по JetOS включает в себя четыре компании:

- **ГосНИИАС / ИСП РАН** – непосредственно разработка ОСРВ
- **ЛаБС (Advalange)** – разработка инструментария сбора и анализа структурного покрытия
- **ИПМ им. Келдыша РАН** – графическая библиотека OpenGL

Задачи верификации в КТ-178С, предполагающие анализ исходного кода

- ***Анализ структурного покрытия*** исходного кода.
  - ***покрытие структурных элементов*** (SEC – Structure Element Coverage)
    - ***Покрытие операторов*** *(SC - Statement Coverage)*
    - ***Покрытие решений*** *(DC - Decision Coverage)*
    - ***Модифицированное покрытие условий/решений*** *(MC/DC - Modified Decision/Condition Coverage)*
  - ***покрытие связности по данным*** *(DCC – Data Coupling Coverage)*
  - ***покрытие связности по управлению*** *(CCC - Control Coupling Coverage)*

- ***Анализ использования памяти в наихудшем случае*** (WCMA - Worst Case Memory Allocation)
  - ***анализ использования стека*** *(WCSA - Worst Case Stack Allocation)*
  - ***анализ использования кучи*** *(WCHA - Worst Case Heap Allocation)*

- ***Анализ времени выполнения в наихудшем случае*** (WCET - Worst-Case Execution Timing)

Существующие решения

	LDRA	Rapita	aiT	Bound-T	VectorCAST	Verocel
Покрытие элементов (SC, DC, MC/DC)	+	+			+	+
Control Coupling	+				+	+-
Data Coupling	+-				+-	
WCET		+	+	+		
WCSA			+	+		+
WCHA						

Проблемы использования существующих инструментов анализа

- *Цена*
- *Функциональность*
- *Ошибки/недоработки в реализации существующих инструментов*

Цель разработки инструментария

		…	…	…	…	…	…
Покрытие элементов (SC, DC, MC/DC)	+	+	+			+	+
Control Coupling	+	+				+	+-
Data Coupling	+	+-				+-	
WCET	+		+	+	+		
WCSA	+			+	+		+
WCHA	+						

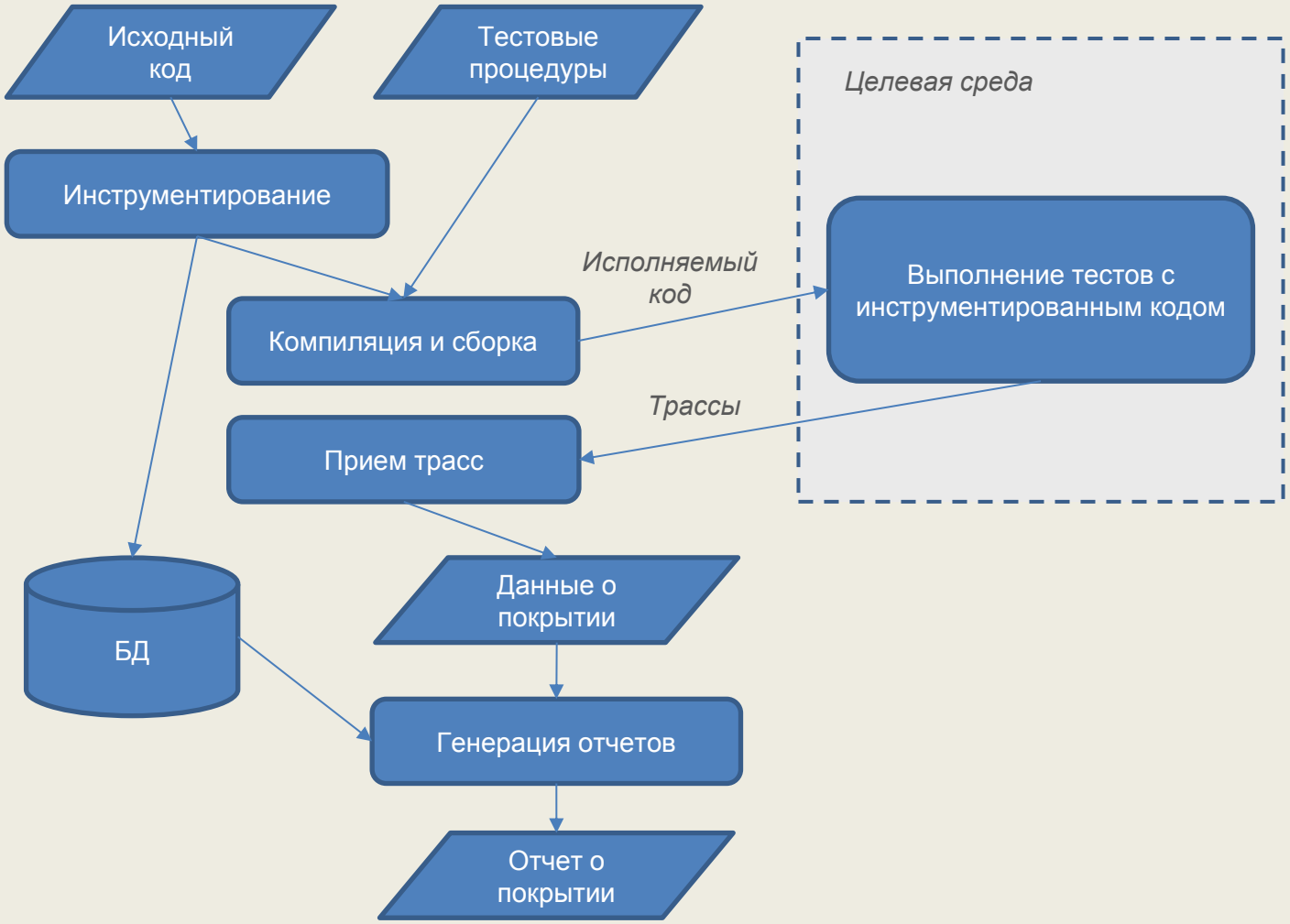
Разрабатываемые инструменты анализа

1. *«E178Cover»* - анализ структурного покрытия исходного кода, включая:
   - покрытие его структурных элементов (операторов и логики) по критериям ***SC, DC, MC/DC***) и
   - покрытие связности компонентов по данным и управлению (***DCC/CCC***)

2. *«E178Time»* - получение оценки времени выполнения в наихудшем случае (***WCET***)

3. *«E178Mem»* - оценка использования динамической памяти: кучи (***WCHA***) и стека (***WCSA***)

Использование инструментов анализа



Инструмент E178Cover – анализ структурного покрытия элементов и связей

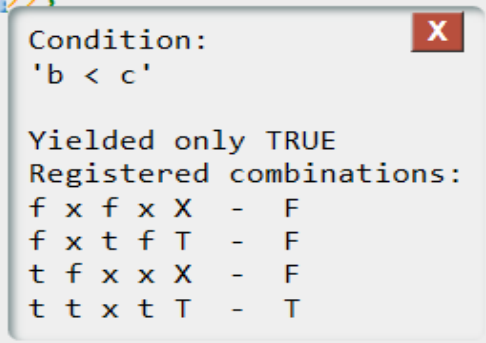
- ✓ Сбор покрытия структурных элементов по критериям SC, DC, MC/DC;
- ✓ Сбор покрытия связности по управлению;
- ✓ Сбор покрытия связности по данным;
- ✓ Формирование отчетов о покрытии (.txt, .html).

Развитие функциональности инструмента анализа структурного покрытия

- *Учет операций сравнения и тернарной операции как логических (MC/DC)*
- *Учет поразрядных операций (MC/DC)*
- *Учет маскирования MC/DC*
- *Учет точек входа и выхода в пределах функции (DC, MC/DC)*
- *Покрытие связности по управлению*
- *Покрытие связности по данным*

MC/DC. Учет операций сравнения и тернарной операции как логических

```
36  static void f_cmp1(int a, int b, int c)
37  {
38      res = (a < b) == (b < c);
39      res = (a < b) != (b < c);
40      res = (a < b) <  (b < c);    // 'b < c' Independent affect not shown
41      res = (a < b) <= (b < c);    // 'a < b' Independent affect not shown
42      res = (a < b) >  (b < c);    // 'a < b' Independent affect not shown
43      res = (a < b) >= (b < c);    // 'b < c' Independent affect not shown
44      res = ((a < b) ? b < c : a < c) && ((a < b) == (b < c));
45  }
46  static void f_cmp2(int a, int b, int c)
47  {
48      res = (a < b) == (b < c);
49      res = (a < b) != (b < c);
50      res = (a < b) <  (b < c);
51      res = (a < b) <= (b < c);
52      res = (a < b) >  (b < c);
53      res = (a < b) >= (b < c);
54      res = ((a < b) ? b < c : a < c) && ((a < b) == (b < c));
55  }
56  void test_cmp()
57  {
58      f_cmp1(1, 2, 3);
59      f_cmp1(1, 2, 2);
60      f_cmp1(2, 2, 3);
61
62      f_cmp2(1, 2, 3);
63      f_cmp2(1, 2, 2);
64      f_cmp2(2, 2, 3);
65      f_cmp2(2, 2, 2);
66  }
```



MC/DC. Учет поразрядных операций

```
4  static void f_bw1(int a, int b)
5  {
6      res = a && b;               // 'a' Yielded only TRUE
7      res = a & b;                // 'a' Yielded only TRUE
8      res = !(a && b);            // 'a' Yielded only TRUE
9      res = !a && !b;             // 'a' Yielded only TRUE, 'b' not covered
10     res = !(a & b);             // 'a' Yielded only TRUE
11     res = !a & !b;              // 'a' Yielded only TRUE, 'b' Independent affect not shown (tF->F, tT->F)
12     res = a || b;               // 'a' Yielded only TRUE, 'b' Not covered
13     res = a | b;                // 'a' Yielded only TRUE, 'b' Independent affect not shown (tF->T, tT->T)
14     res = a ^ b;                // 'a' Yielded only TRUE
15     res = a && (a ^ b);         // 'a' Yielded only TRUE
16     res = (a & b) == (a ^ b);   // 'b' Independent affect not shown (tftF->F, tttT->F)
17 }
18 static void f_bw2(int a, int b)
19 {
20     res = a && (a || b);        // 'a' Yielded only TRUE, 'b' not covered
21     res = a && (a && b);        // 'a' Yielded only TRUE
22     res = a && (a ^ b);         // 'a' Yielded only TRUE
23     res = (a & b) == (a ^ b);   // 'a', 'b' covered by Masking MCDC criterion
24 }
25 void test_bw()
26 {
27     f_bw1(1, 0);
28     f_bw1(1, 1);
29
30     f_bw2(0, 0);
31     f_bw2(0, 1);
32     f_bw2(1, 0);
33     f_bw2(1, 1);
34 }
```

MC/DC. Учет маскирования

```
68  void f_msk(int a, int b, int c)
69  {
70      // covered by Masking MCDC criterion
71      res = (a & b) == (a ^ b);
72      res = (a ? b : c) && (b ? a : c);
73      res = (a < b) == (((a < b) <  (b < c)) != ((a < b) >  (b < c)));
74      res = ((a < b) <  (b < c)) != ((a < b) > (b < c));
75      res = (a && b) || (!a && c);
76      res = (a && b) || (a && c);
77  }
78  void test_masking()
79  {
80      f_msk(0,0,0);
81      f_msk(0,0,1);
82      f_msk(0,1,0);
83      f_msk(0,1,1);
84      f_msk(1,0,0);
85      f_msk(1,0,1);
86      f_msk(1,1,0);
87      f_msk(1,1,1);
88  }
```

Condition:
'a < b'

covered by Masking MCDC criterion

DC, MC/DC. Покрытие точек входа/выхода

```
 6      a = 1;
 7
 8      switch (a)
 9      {
10          case 1:
11          case 2:
12              a=a;
13      }
14
15      switch (a)
16      {
17      case 2:
18          a = a;
19      }
20
21      switch (a)
22      {
23          default:
24          case 1:
25              a=a;
26      }
27
28      switch (a)
29      L: case 1: case 2: default: a=a;
30
```

Покрытие связности по управлению

Control Coupling coverage report *generated on 09.06.2019 18:46:16*

E178Cover tool ver. 1.3.1

#	File	Covered	Partially covered	Not covered	Unexpected	Outbound	Inbound
1. Directory d:\E178_Cover\demo\example4\src							
1	F0.c	15	0	3	2	20	6
2	F1.c	1	0	0	0	1	17

#	Function	Covered	Partially covered	Not covered	Unexpected	Outbound	Inbound
1	int f11(int)	0	0	0	0	0	5
2	FPTR getFx(int)	0	0	0	0	0	4
3	int f12(int)	1	0	0	0	1	3

#	Call	Callee	Coverage Info
1	F1.c(10:12)f11(a)	F1.c(3:1)int f11(int)	COVERED

#	Inbound link	Coverage Info
1	F0.c(6:12)int fp(FPTR)::(p)(1)	NOT COVERED
2	F0.c(23:5)int fmain()::(pf1)(1)	NOT COVERED
3	F2.c(5:23)void f21(int,int)::f12(b)	COVERED

#	Function	Covered	Partially covered	Not covered	Unexpected	Outbound	Inbound
4	int f13(int)	0	0	0	0	0	2
5	long f14(int)	0	0	0	0	0	1
6	void f15(int)	0	0	0	0	0	1
7	void f16(int)	0	0	0	0	0	1

3	F2.c	3	0	0	0	3	1
Overall Summary		19	0	3	2	24	

Покрытие связности по управлению

Control Coupling coverage report *generated on 09.06.2019 18:46:16*

E178Cover tool ver. 1.3.1

F0.c

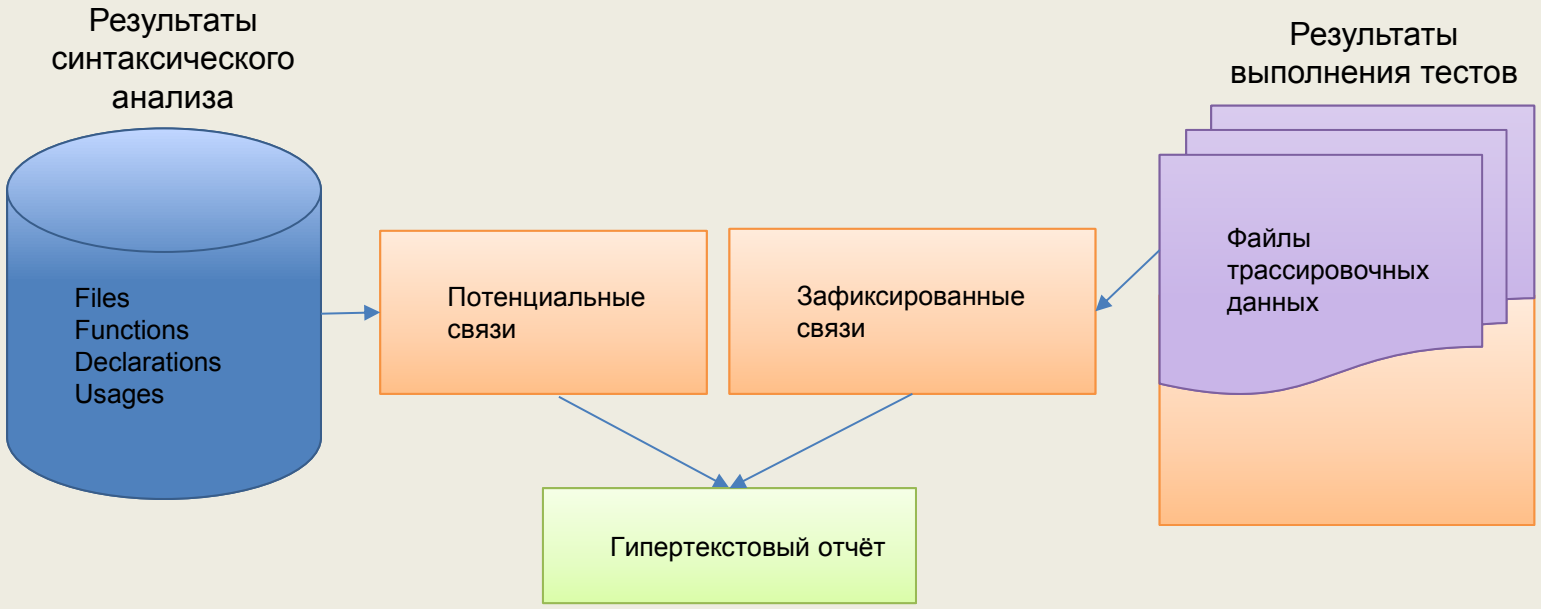
```
1  + #include "F1.h"
2  + #include "F2.h"
3
4  int fp(FPTR p)
5  {
6      return (p)(1);
7  }
8
9  void f0()
10 {
11     f11(1);
12     f21(1, 2);
13 }
14
15 int fmain()
16 {
17     int(*pf1)(int);
18
19     pf1 = &f11;
20
21     f0();
22
23     (pf1)(1);
24
25     fp(&f11);
26
27     fp(getFx(1));
28     fp(getFx(3));
29     fp(getFx(4));
30     fp(getFx(5));
31
32     return 0;
33 }
```

<< To the first | < Previous | Next > | To the last >>

Покрытие связности по данным

1. Связи по разделяемым данным
2. Связи по параметрам
3. Связи по внешним данным

Формирование отчета о покрытии связности



Покрытие связности по разделяемым данным

Data Coupling coverage report *generated on 12.10.2018 15:46:10* — E178Cover tool ver. 1.2.3

#	File	Outbound	Inbound	Unused data	Undefined data
1. Directory d:\E178_Cover\Installer\demo\example_DCC\exampleDCC3\src					
1	F0.c	5 | 1	8 | 2	2	2

#	Function	Outbound	Inbound	Unused data	Undefined data
1	void fset(int)	3 | 0	0	0	0
2	int f0(int)	0	6 | 2	1	2
3	void f1()	2 | 1	2 | 0	1	0

#	Link To	Data definition	Def	Use	Coverage Info
1	F0.c(24:1) int f0(int)	F0.c(13:1) static int aaa	F0.c(35:5) aaa++	F0.c(30:12) return aaa	NOT COVERED
2	F0.c(24:1) int f0(int)	F2.c(10:1) int gi2 = -2	F0.c(36:5) gi2 = f0(gi1)	F0.c(26:21) gi0 ? gi1 : gi2	COVERED

#	Link From	Data definition	Def	Use	Coverage Info
1	F0.c(18:1) void fset(int)	F0.c(13:1) static int aaa	F0.c(20:5) aaa = v	F0.c(35:5) aaa++	COVERED
2	F1.c(14:1) int f11(int)	F0.c(13:1) static int aaa	F1.c(16:5) gi1 = 1	F0.c(36:14) f0(gi1)	COVERED

#	Variable	Data definition	Data access point	Error
1	aaa	F0.c(13:1) static int aaa	F0.c(35:5) aaa++	NOT USED

2	F1.c	2 | 1	0	0	0
3	F2.c	1 | 0	0	0	0
Overall Summary		Total 8 | Not covered 2		2	2

Покрытие связности по разделяемым данным

Data Coupling coverage report *generated on 12.10.2018 15:46:10*

E178Cover tool ver. 1.2.3

F0.c

```
1  - #include "F1.h"
     --- file d:\E178_Cover\Installer\demo\example_DCC\exampleDCC3\src/F1.h --- (nesting level 1)
1  #ifndef XXX1
2  #define XXX1
3  int f11(int a);
4  int f12(int a);
5  #endif
2  + #include "F2.h"

3
4  static int aaa;
5  int gi0 = -1;
6  extern int gi1;
7  extern int gi2;
8
9  static void fset(int v)
10 {
11     aaa = v;
12     gi0 = 0;
13 }
14
15 int f0(i
16 {
17     a = gi0 ? gi1 : gi2;
18     fset(1);
19     gi2 = 0;
20     int x1 = f11(1), b;
21     return aaa;
22 }
23
24 void f1()
25 {
26     aaa++;
27     gi2 = f0(g
28 }
```

Data read point:
- DCC links: total 2 | covered 2

Data read point:
- DCC links: total 2 | covered 1

Inbound DCC links:
- [covered] from component: 'void fset(int)'
  source: (F0.c : 20) 'aaa = v'
- [not covered] from component: 'void f1()'
  source: (F0.c : 35) 'aaa++'

<< To the first | < Previous | Next > | To the last >>

Спасибо за внимание!