2013 CEE-SEC(R) Software Engineering Conference in Russia

Интегрированная технология верификации и тестирования промышленного программного продукта

Всеволод Котляров
Павел Дробинцев
vpk@spbstu.ru
Санкт-Петербургский государственный политехнический университет
Институт информационных технологий и управления, каф. ИУС

Важные свойства производства промышленного программного продукта

В настоящее время в связи с ростом мощности аппаратных и программных средств в списке дефицитных ресурсов разработки программного продукта (ПП) первые места заняли трудоемкость, надежность и эффективность. Трудоемкость определяет необходимый объем инвестиций и сроки производства ПП, надежность – защищенность ПП от сбоев и дефектов, а эффективность – пропускную способность обработки информации.

Минимизация трудоемкости за счет автоматизации производства ПП

В настоящее время появились технологии и средства автоматизации разработки ПП обеспечивающие почти полностью автоматизированный цикл производства [Rational Rose, VRS/TAT].

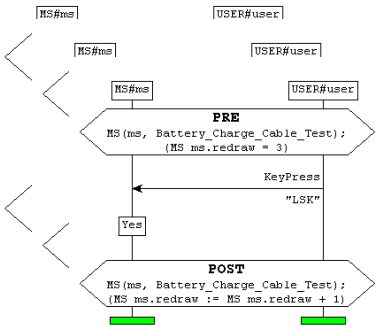
Например в технологии VRS/TAT по формализованным спецификациям на систему генерируется формализованная поведенческая модель, корректность которой доказывается, а затем по корректной модели генерируется код ПП и символьные сценарии, гарантирующие полноту экспериментальной проверки поведения разрабатываемого ПП относительно выбранных критериев

Технологическая цепочка

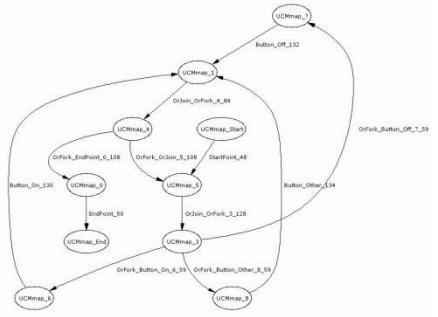
Неформальные требования на естественном языку

	4	6	8
1		8.2.2 Contact list Management	
2	Req Id	Source Text of the Title, Requirement, or Comment	Scenario of requirement verifying
290	TRS_18081-2210[7]	*See Section "Self Presence Reporting" and Section "Silent Mode Settings".*	
291	TRS_18081-2209[6]	When My Availability is set to Do Not Disturb, the user's presence status is reported to the server as Do Not Disturb.	scen#64 + 1. Select the option 'Do Not Disturb' in the KPTT_MyStatus menu and press the LSK 'Select' or the HK 'Center Select'. 2. Check that the KPTT_Confirmation_Notice menu (body: My Availability changed to Do Not Disturb) appeared
292	TRS_18081-2211[7]	*See Section "Self Presence Reporting".*	
293	TRS_18081-2212[7]	When My Availability is set to Do Not Disturb, the user's ability to originate PTT calls shall not be affected.	1. Check that state of the handset is DND 2. Go to the KPTT_Contacts menu, select a contact (it's own state shall be not DND - for more information see 8.2.2.1.1 Individual Contact Presence) 3. Press the PTT button --- manual test: Check that a PTT session starts

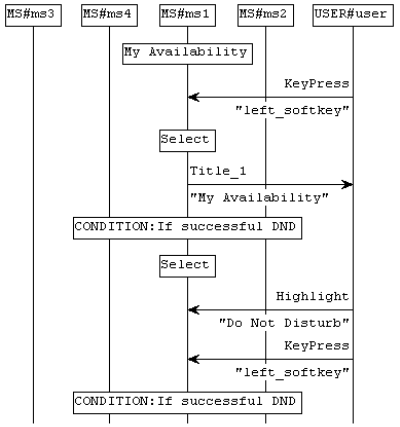
Формализованные требования



Поведенческая модель приложения в виде машины состояний для генерации кода для заданной платформы



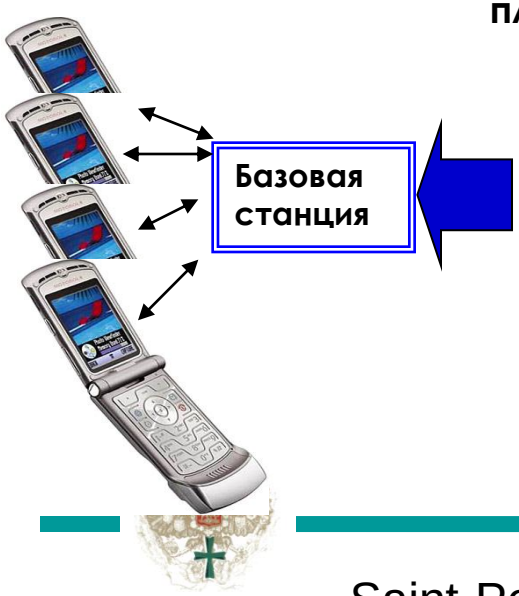
Тестовые сценарии на языке MSC для генерации кода тестов



Тесты и приложение на языке целевой платформы (C/C++, Java, Pearl)

```
386 mobilePhone 1.display_waitForTextOnDisplayIgnoreSpace('PTT Message')
387 #224
388 mobilePhone 1.display_waitForTextOnDisplayIgnoreSpace('name3 has joined group1';5)
389 #225
390 mobilePhone 1.keyPress_pushButtons('LSK')
391 #226
392 mobilePhone 1.display_waitForTextOnDisplayIgnoreSpace('PTT Contacts')
393 #227
394 NOT mobilePhone 1.display_waitForTextOnDisplayIgnoreSpace(name1;5)
395
396 #Log:++++++++++++++++++++++++++++++++++++++++++++++++++Stopped++++++++++++++++++++++++++++++++
397 general.DialogBoxYES_No('(Finding: Label 1)')
398 #Log:++++++++++++++++++++++++++++++++++++++++++++++++++Resumed++++++++++++++++++++++++++++++++
399
400 #231
401 mobilePhone 1.keyPress_scrollToText(group1)
402 mobilePhone 1.display_verifyHighlightedText(group1)
403 #232
404 mobilePhone 1.keyPress_pushButtons('LSK')
405 #233
406 mobilePhone 1.keyPress_scrollToText('Add Member')
407 mobilePhone 1.display_verifyHighlightedText('Add Member')
408 #234
409 mobilePhone 1.keyPress_pushButtons('LSK')
410 #235
411 mobilePhone 1.display_waitForTextOnDisplayIgnoreSpace('Group Members')
412 #236
413 NOT mobilePhone 1.display_waitForTextOnDisplayIgnoreSpace(name1;5)
414 #237
415 mobilePhone 1.display_waitForTextOnDisplayIgnoreSpace(name2;5)
```

Исполнение тестов



Проблема формализации требований

В практике промышленной разработке программного проекта:

- **В проектной документации формулировка требований задается либо конструктивно, когда из текста требования на естественном языке удается реконструировать процедуру контроля или сценарий проверки выполнения данного требования, либо неконструктивно, когда заданное свойство, не содержит пояснения способа его проверки**
- **Процедура проверки требования – это точная последовательность причин и следствий некоторых активностей (кодируемых действиями, сигналами, состояниями), в результате реализации которой можно утверждать, что данное требование выполнено - т.н. критериальная процедура**
- **Отслеживая в поведенческом сценарии системы (гипотетическом, реализованном в модели или в реальной системе) факты выполнения критериальной процедуры, можно утверждать, что соответствующее требование в анализируемой системе удовлетворено**

Источники недоопределенности требований и спецификаций на этапе проектирования

- **Несогласованность поведенческой логики в требованиях и спецификациях**
- **Недоучет областей корректной функциональности программного изделия**
- **Недоучет взаимодействия процессов на общих ресурсах и протоколах**
- **Недоучет временных ограничений**

Решение проблемы во внедрении формального подхода и методов доказательного проектирования в практику разработки качественного программного продукта

Инженерная нотация – ключ к применимости формальных методов в промышленности

С.С.Лавров: «...Применимость формальных методов в промышленности в огромной степени определяется тем, насколько адекватен язык формализации принятой инженерной практике, в которую вовлечены не только разработчики и тестировщики, но и заказчики, руководители проектов разных уровней, маркетологи и другие специалисты.»

Базовый протокол: $\forall x(\alpha \rightarrow < u > \beta)$

Поведенческая трасса:

$$l_1 : s_1 \xrightarrow{a_1} l_2 : s_2 \xrightarrow{a_2} \ldots \xrightarrow{a_{n-1}} l_n : s_n$$

$$l_1 : s_1 \xrightarrow{B_1} l_2 : s_2 \xrightarrow{B_2} \ldots \xrightarrow{B_{n-1}} l_n : s_n$$

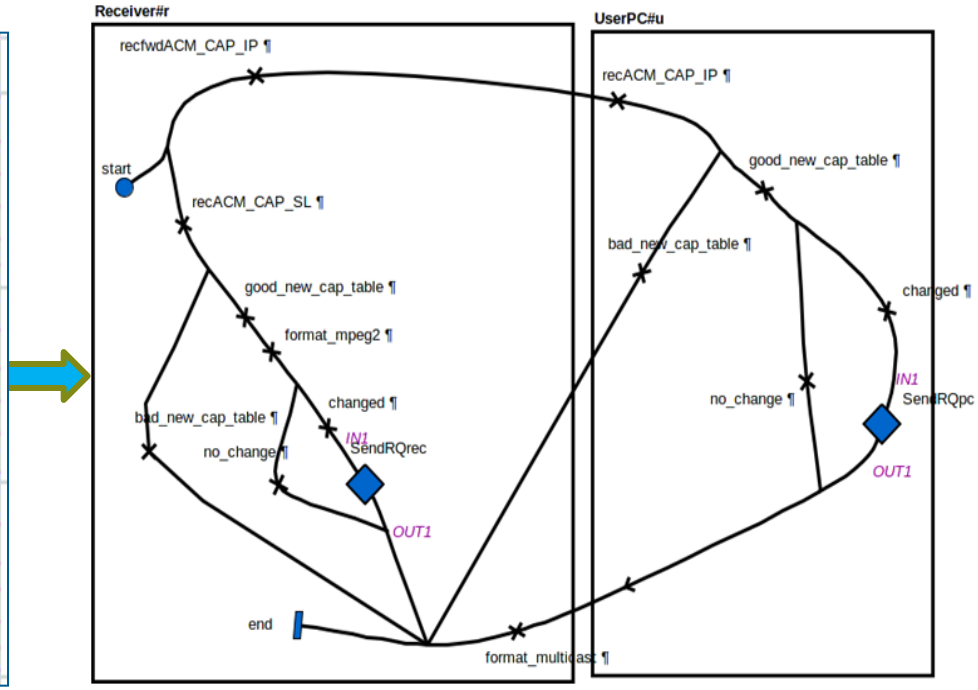
$S_1, S_2, \ldots S_n$ – промежуточные состояния

$B_1, B_2, \ldots B_{n-1}$ – базовые протоколы

$l_1, l_2, \ldots l_n$ – метки

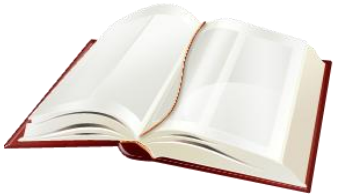
От неформальных спецификаций к формальной модели

Identifier	Requirements	Traceability	Traces
FREQ_GWR.1	Gateway shall transmit ACM_CAP messages repeatedly at an interval of T1 (TBD). This message will carry the ACM Capabilities table.	FREQ_GWR.3	
FREQ_GWR.2	Upon a change in the ACM Capabilities table, the Gateway shall send a new version of ACM_CAP message to the Satellite Terminal.	FREQ_GWR.3	
FREQ_GWR.3	Depending on configuration, ACM_CAP message shall be transmitted either in the MPEG2 private section or in a Multicast message sent across the satellite link.	recACM_CAP_SL recfwdACM_CAP_IP recACM_CAP_IP	FRED_GWR_3-1 FRED_GWR_3-2

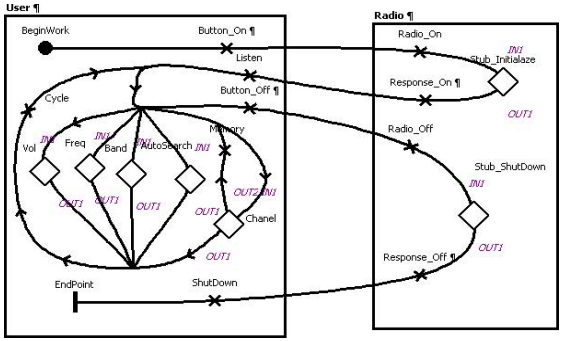


- **Текстовые требования на естественном языке должны быть спроектированы на формальную поведенческую модель, отображающую потоки управления и данных, и наоборот**
- **Все этапы разработки должны быть контролируемы на уровне формальной модели**

Парадигма технологии: создание формальных моделей ПП, доказательство их корректности, тестирование и генерация кода



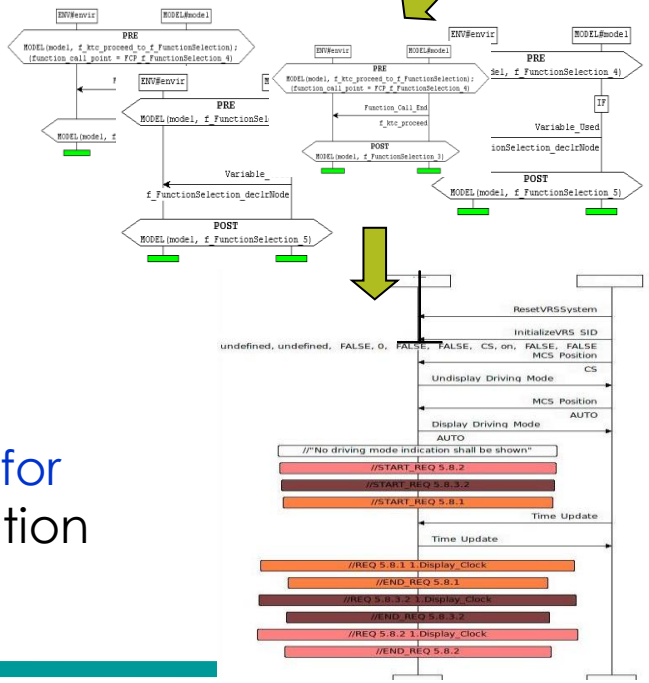
Source Unformal Requarements

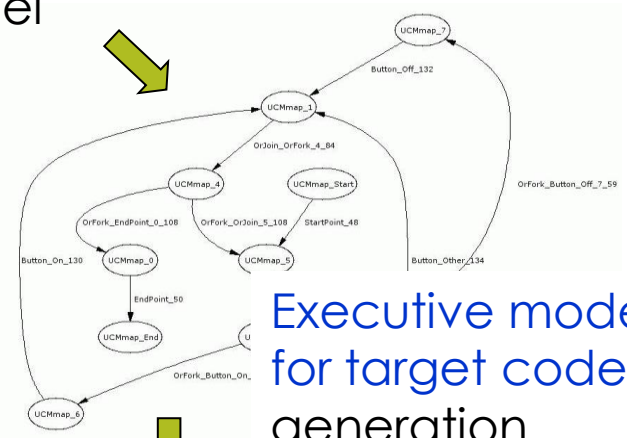


UCM application model

Requirement specification formal behaviour model represented from scenarios and events

Formal BP model for correctness behavior proving



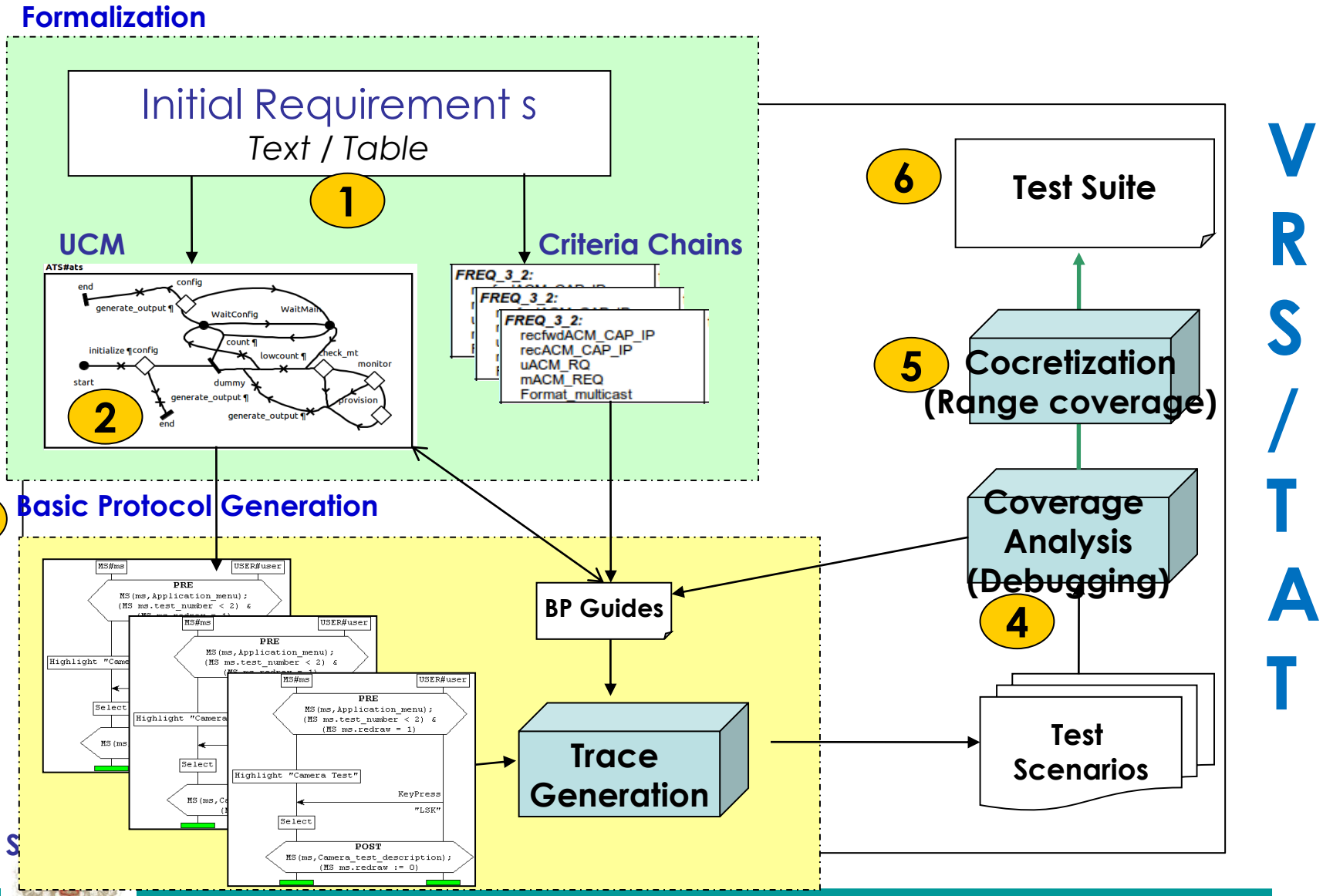


Executive model for target code generation

Executive model for test suites generation

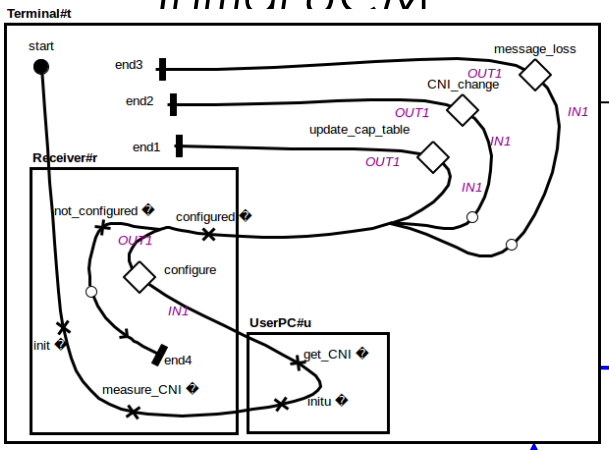
System of testing automation and defect analysis

Генерация сценариев и тестов

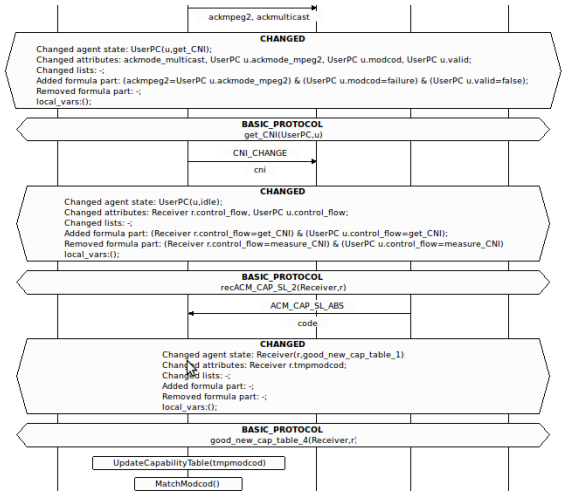


Анализ и корректировка сценариев

Initial UCM



Trace Generation

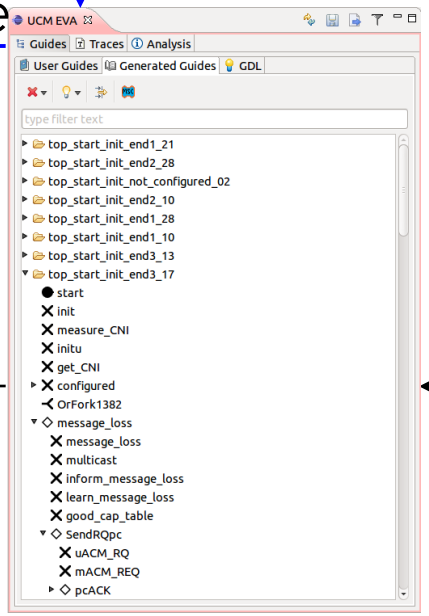


Test Scenarios

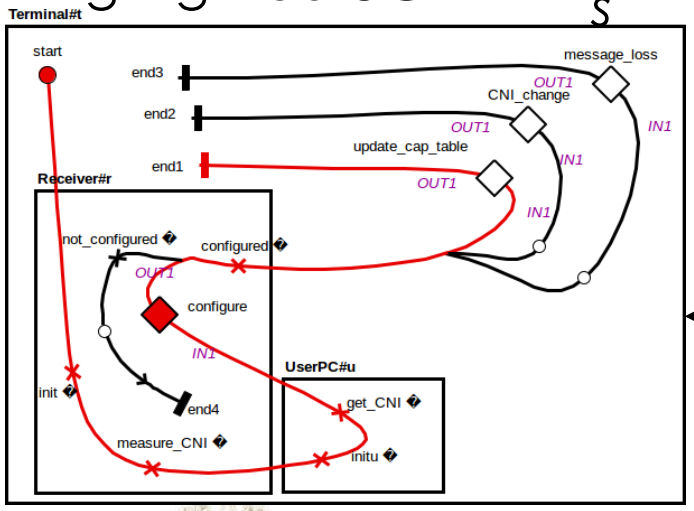
Analysis

Changes

EVA



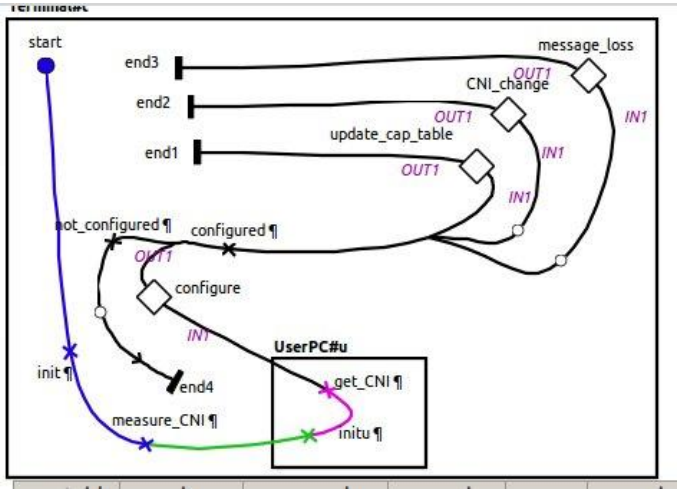
Highlighted UCM



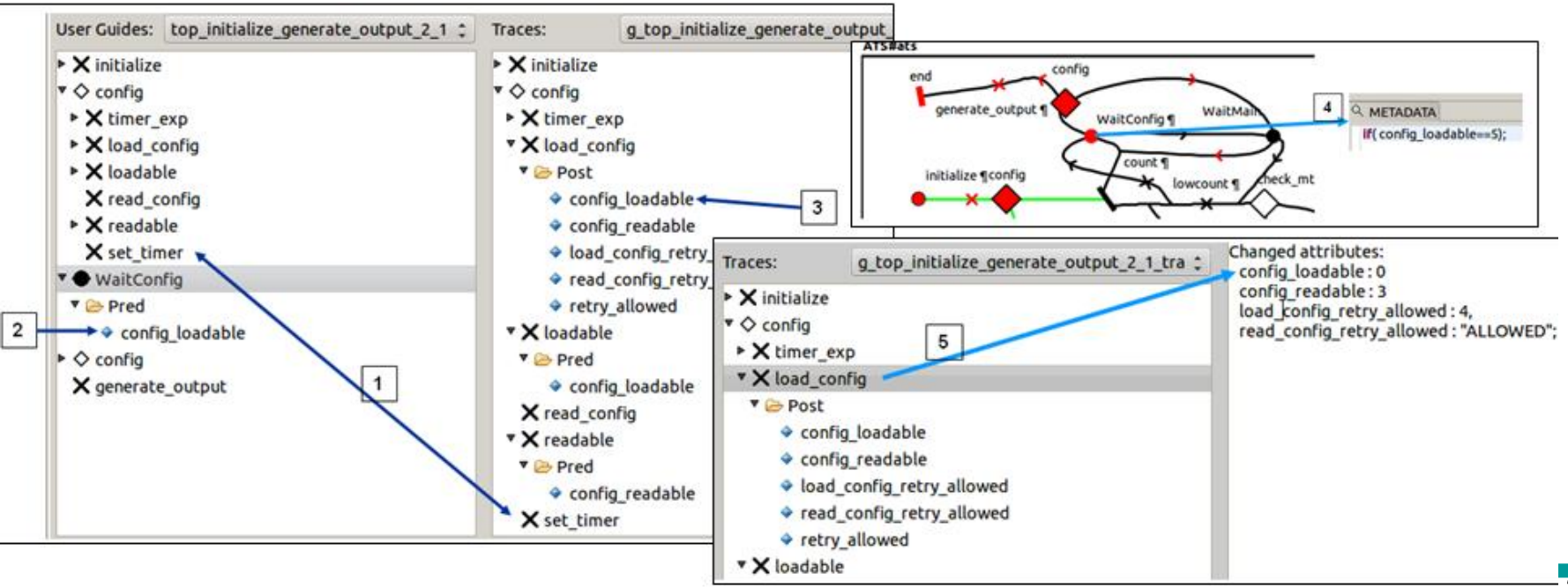
Import to EVA

Пример отладки сценариев

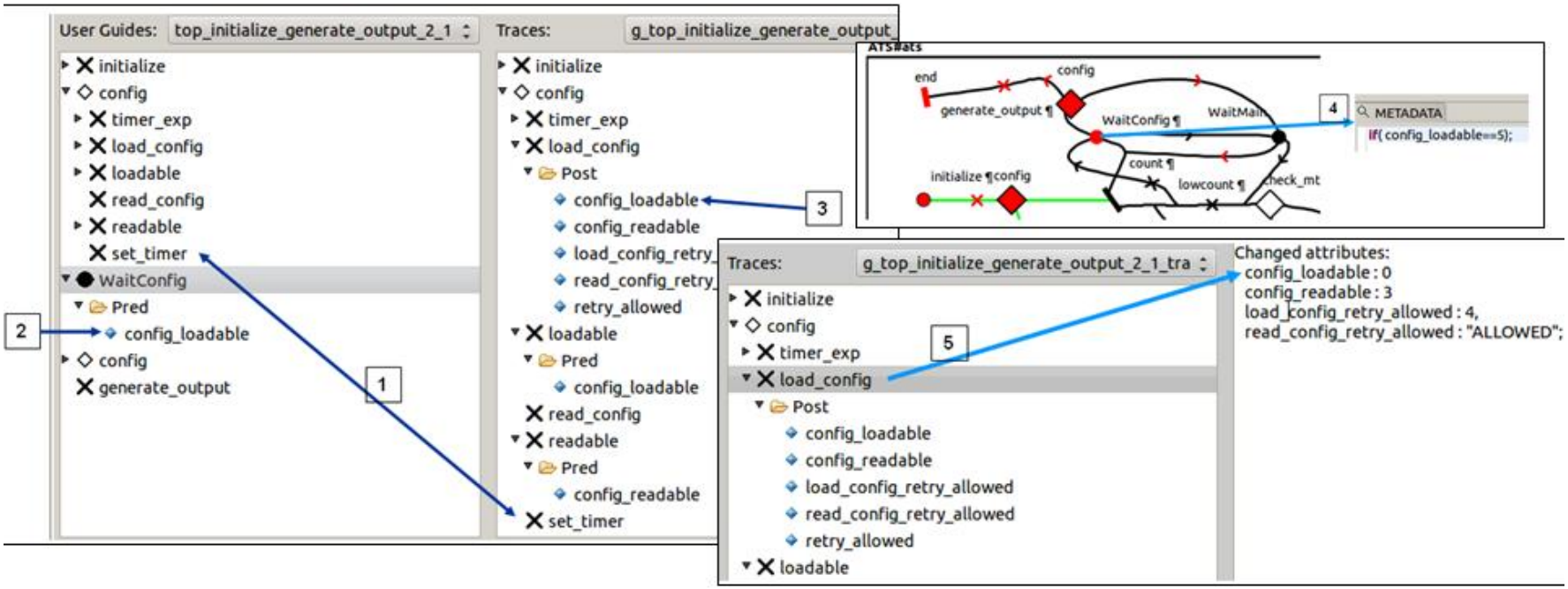
Interactive Analysis
based on UCM element



Static Analysis

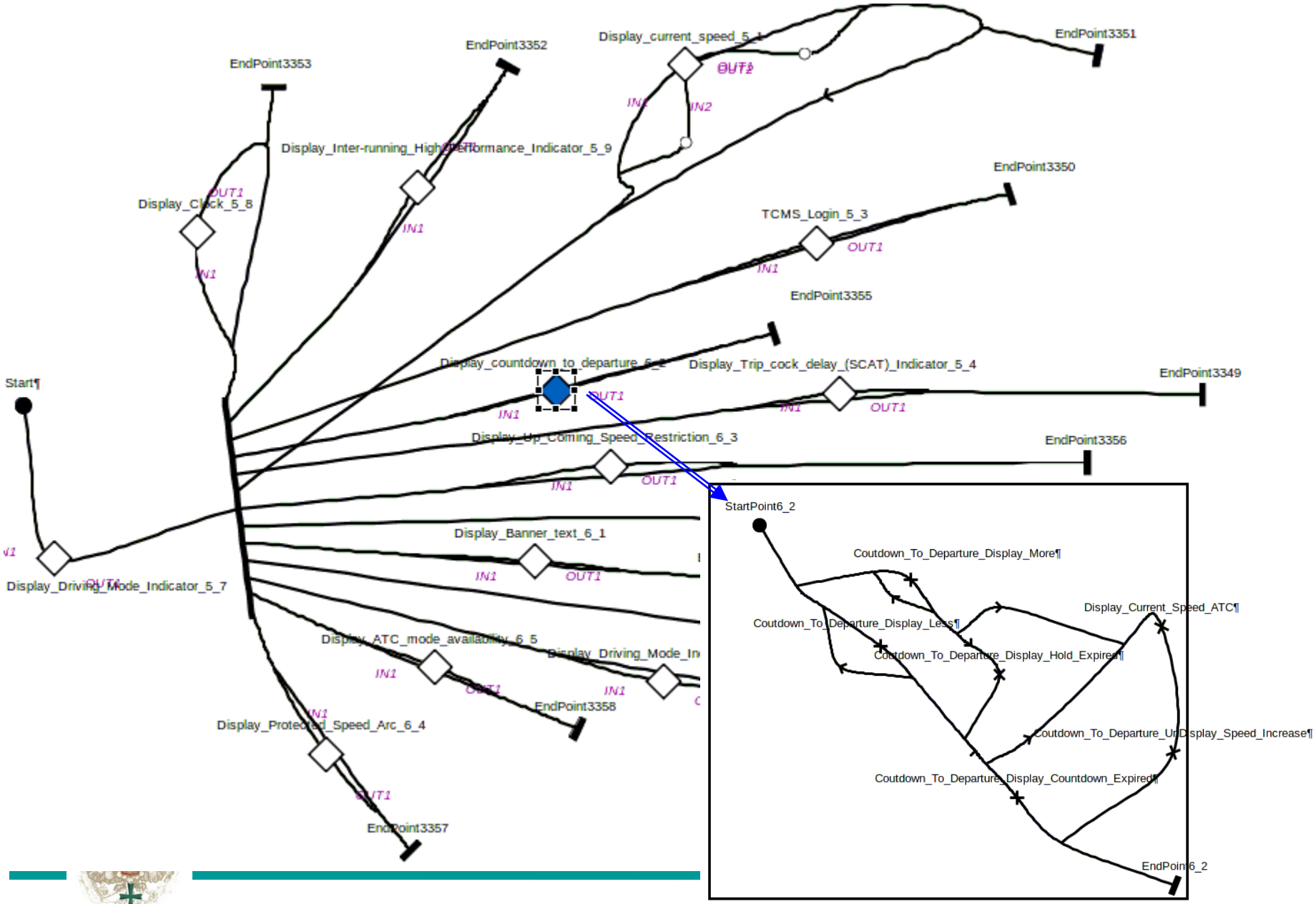


Статический анализ



1. **Найти эдемент расхождения гила и трассы**
2. **Проанализировать предусловие следующего за расхождением элемента гида и определить переменные, влияющие на генерацию трассы**
3. **Найти места изменений значений иди сравнений этих переменных**
4. **Анализировать величины значений переменных в UCM ‘элементе и трассе**

Пример сложного сценария



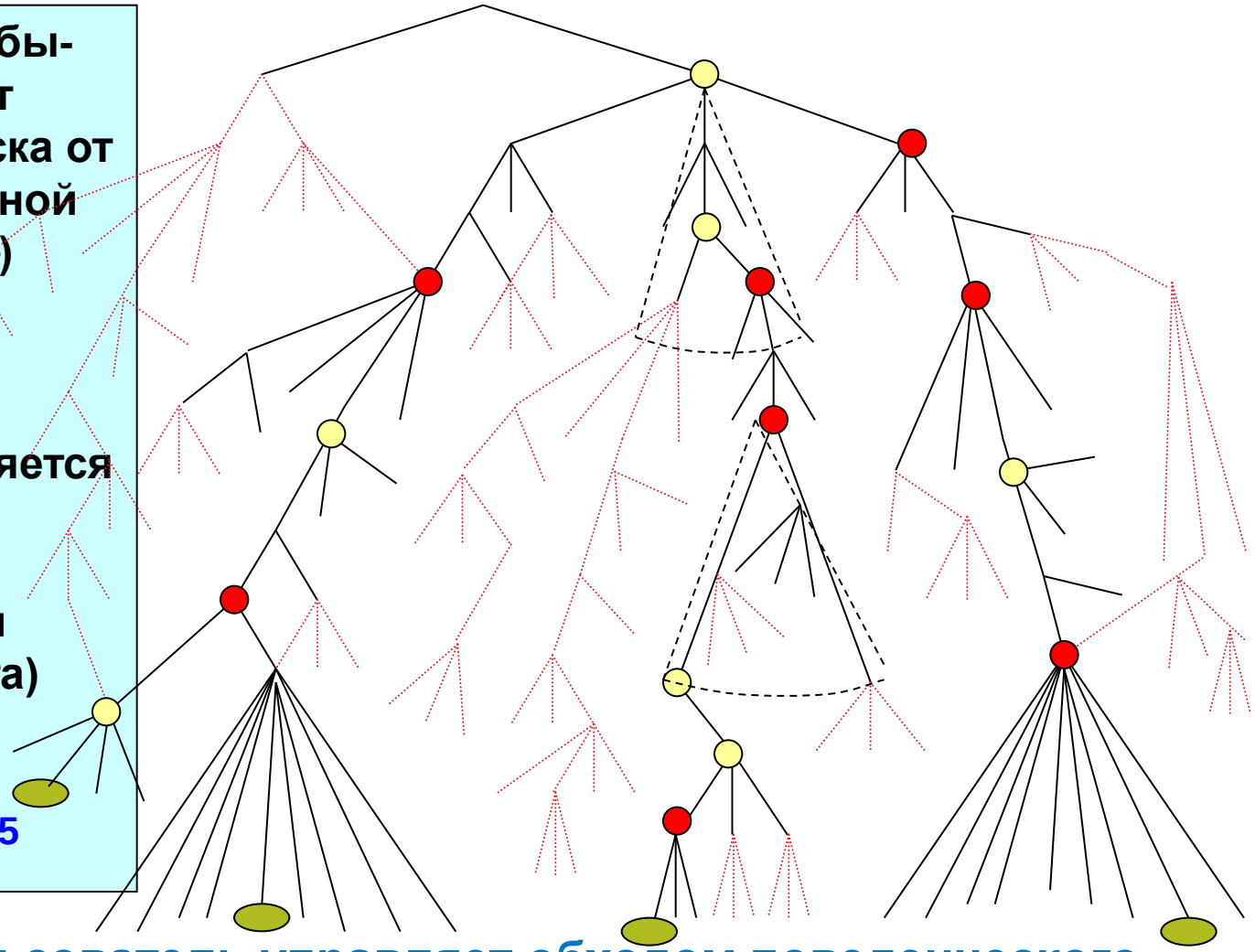
Комбинаторный взрыв – основная проблема анализа промышленных проектов

Решение в сокращении поведенческого пространства введением гидов, учитывающих проблемную область

Гиды – цепочки событий - ограничивают пространство поиска от одной промежуточной цели до другой ( )

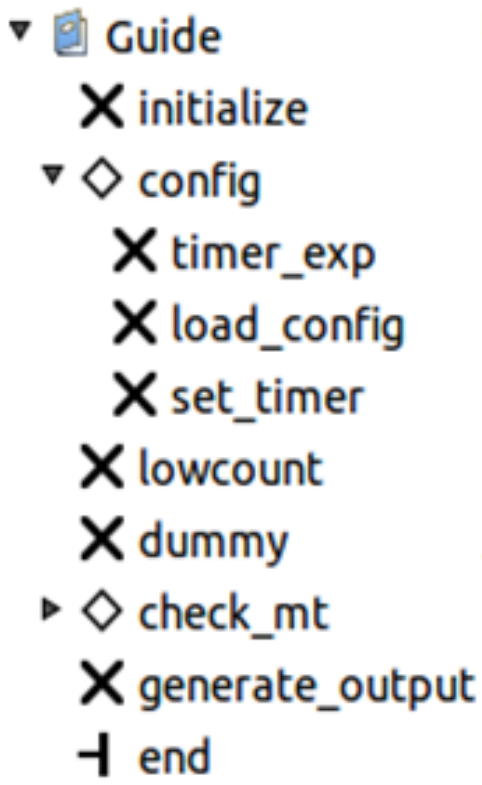
Путь от начала до конечной точки сценариев дополняется промежуточными точками ( ) (формулируемыми экспертами проекта)

Результат СМ РТТ проекта: $2^{150} => 2^{5}$



В VRS пользователь управляет обходом поведенческого пространства с помошью гидов

Автоматизация создания гидов

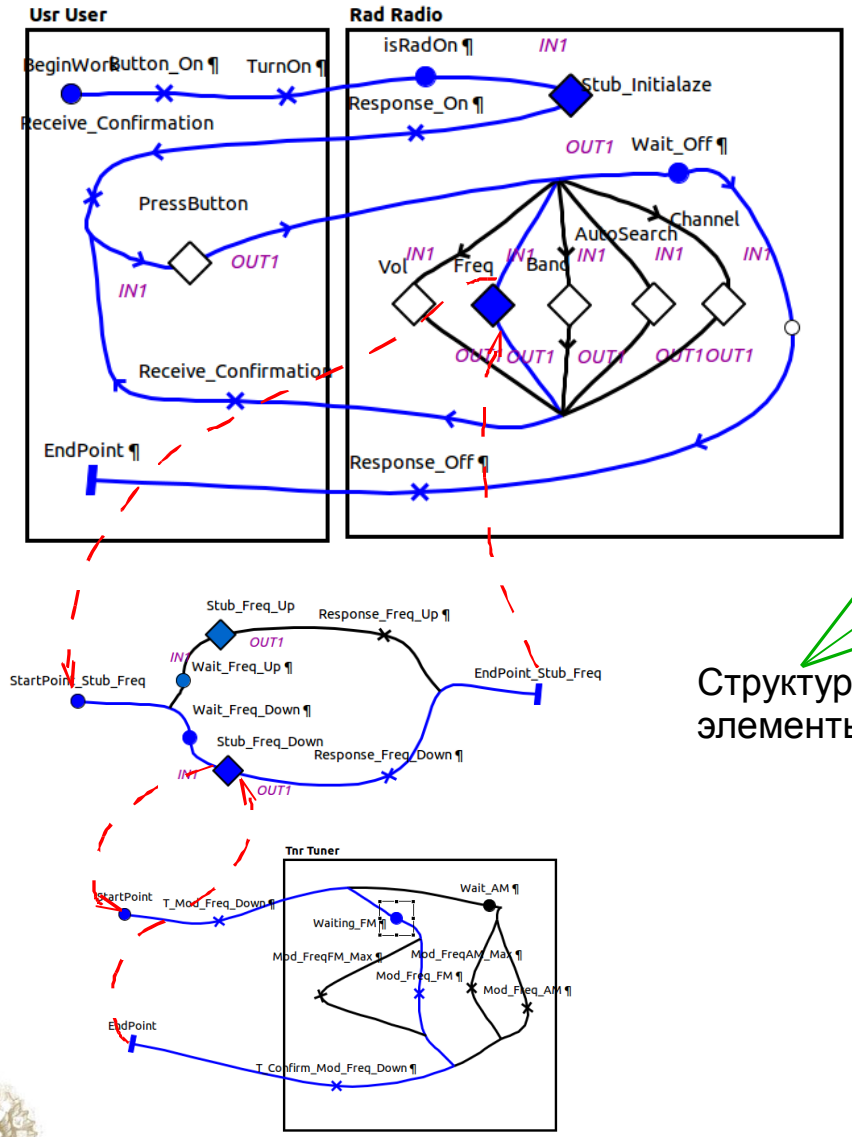



```
patterns: obj(
    p_Guide:(
        ats_0(initialize#1);
        ats_0(timer_exp_1#1);
        ats_0(load_config_1#1);
        ats_0(set_timer_1#1);
        ats_1(lowcount#1);
        ats_1(dummy#1);
        ats_1(report_req_SADM#1);
        ats_2(generate_output_2#1)
    )
);
guides: obj(
    Number_Of_Traces:1;
    g_Guide:(
        pattern:(p_Guide)
    )
);
```

На первом этапе создаются гиды, обеспечивающих заданные критерии покрытия поведения системы.

На втором этапе гиды на языке базовых протоколов и под управлением гидов производится генерация трасс

Структуризация гидов

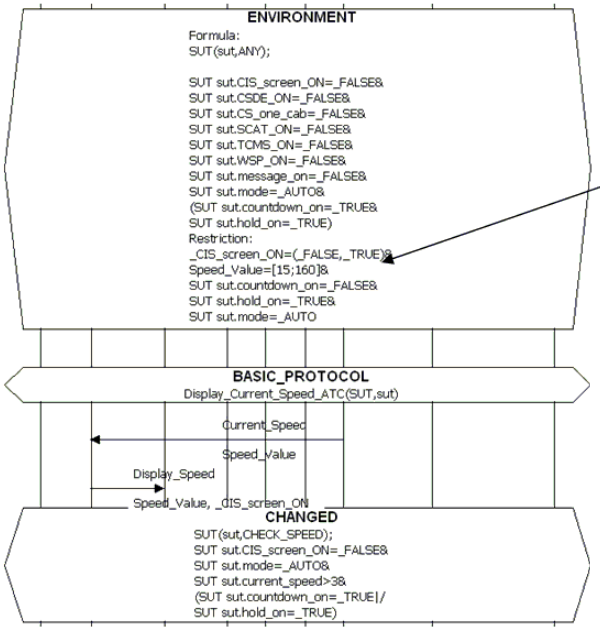


Структура гида по элементам Stub

Структурные элементы Stub

- Button_On_TO_EndPoint_01
  - Button_On
  - TurnOn
  - isRadOn
  - Stub_Initialaze
  - Response_On
  - Receive_Confirmation
  - Freq
    - Wait_Freq_Down
    - Stub_Freq_Down
      - T_Mod_Freq_Down
      - Waiting_FM
      - Mod_Freq_FM
      - T_Confirm_Mod_Freq_Down
      - EndPoint
    - Response_Freq_Down
  - Receive_Confirmation
  - Wait_Off
  - Response_Off
  - EndPoint

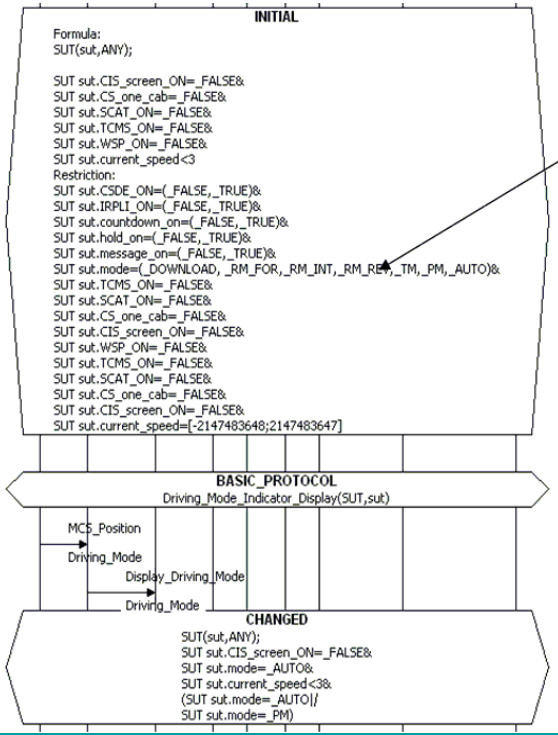
Символические трассы с областями допустимых значений переменных



Speed_Value=[15, 160]

Interesting points:
Speed_Value = 15
Speed_Value = 160
Speed_Value = 73
Speed_Value = 14
Speed_Value = 161

Фрагмент пучка трасс на основе символьной трассы для переменной типа enumeration



Driving_mode=[_DOWNLOAD, _RM_FOR,_RM_INT,_RM_REV,_TM,_PM,_AUTO]

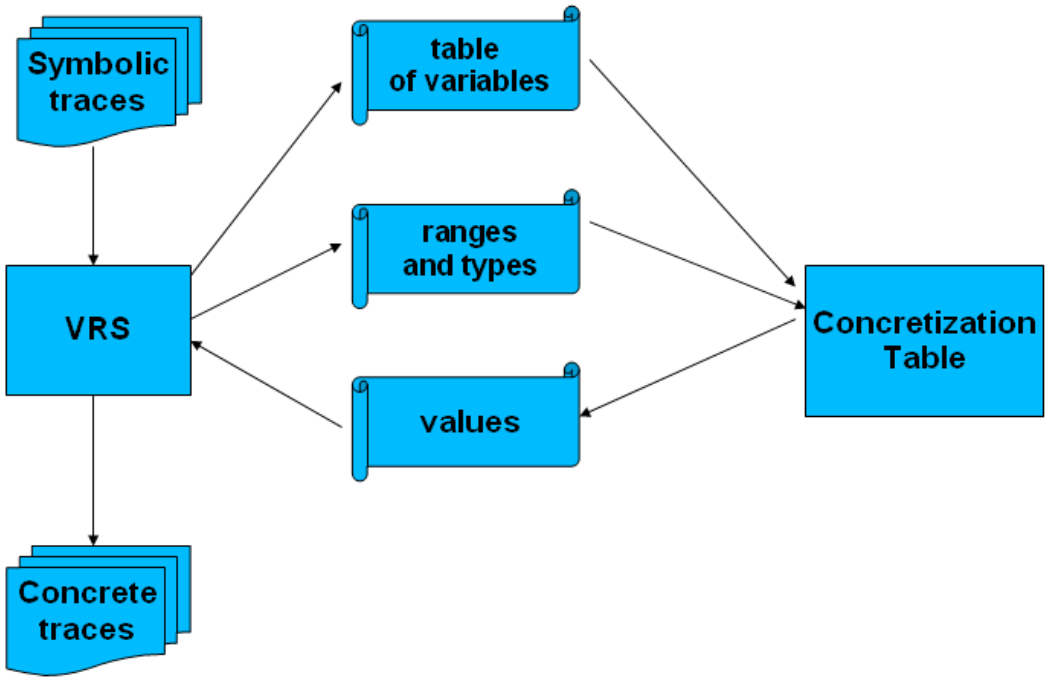
This modes of system applicable for the particular symbolic trace.

Interesting points:
mode =_DOWNLOAD
mode = _qwerty

Фрагмент пучка трасс на основе символьной трассы для переменной типа integer

Конкретизация

Инструмент конкретизации состоит из двух взаимодкйствующих модулей Concretizator и Substitutor. Инструмент по набору символьных трасс создает таблицу конкретизации, в которой заполняет столбцы имен переменных и разрешенных диапазонов значений (Range), а затем обходит каждую трассу и обращается к Substitutor за конкретным значением встреченной в трассе переменной. Substitutor на основе плана конкретизации вычисляет по Range подставляемое значение и возвращает его в Concretizator.



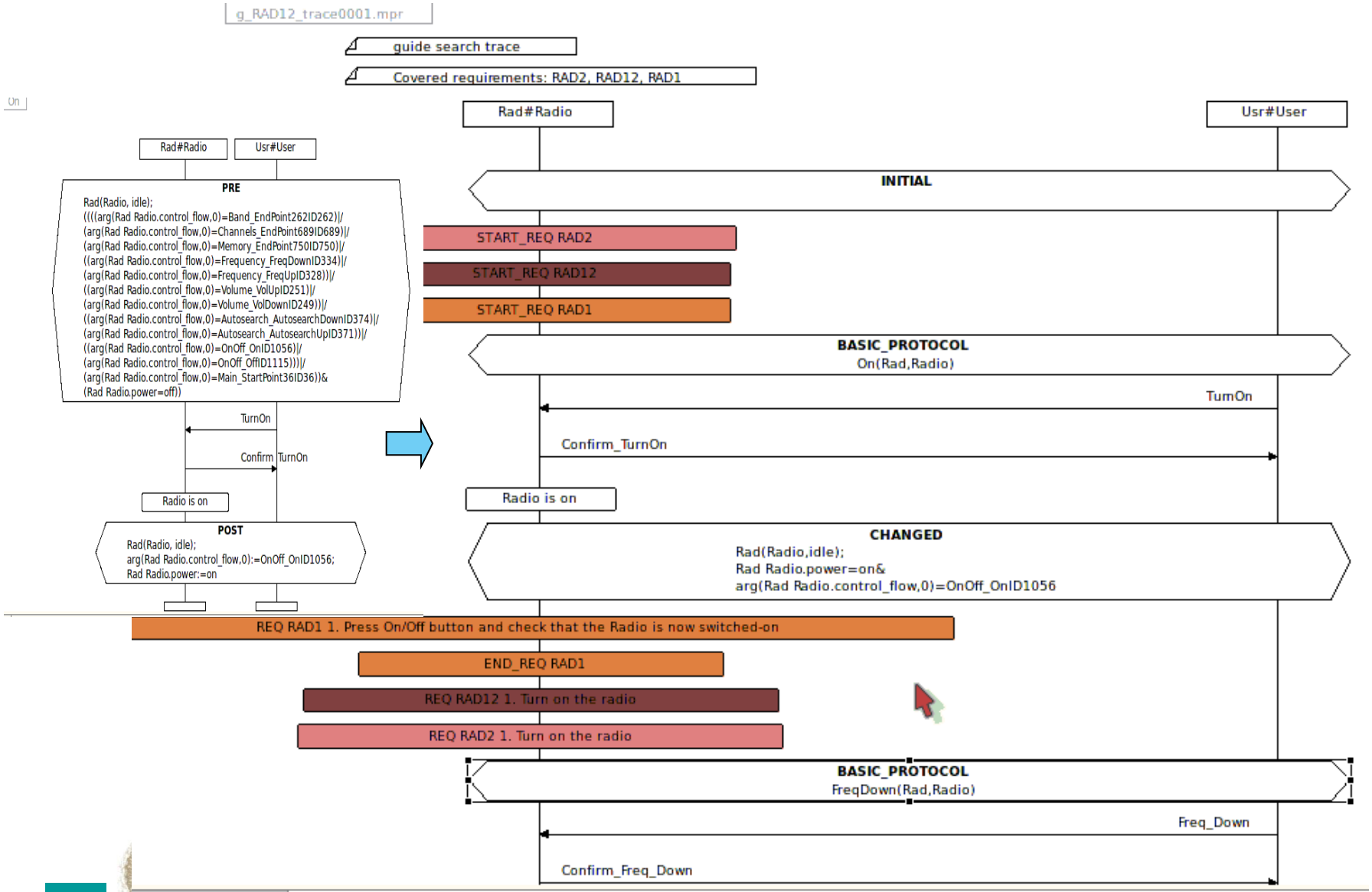
Конкретизация

План автоматической конкретизации тестового набора заносится в колонку User option и формируется в терминах команд управления: R (правая граница Range),M (середина),L (левая),O (значение за границей),C (конкретная величина из Range)

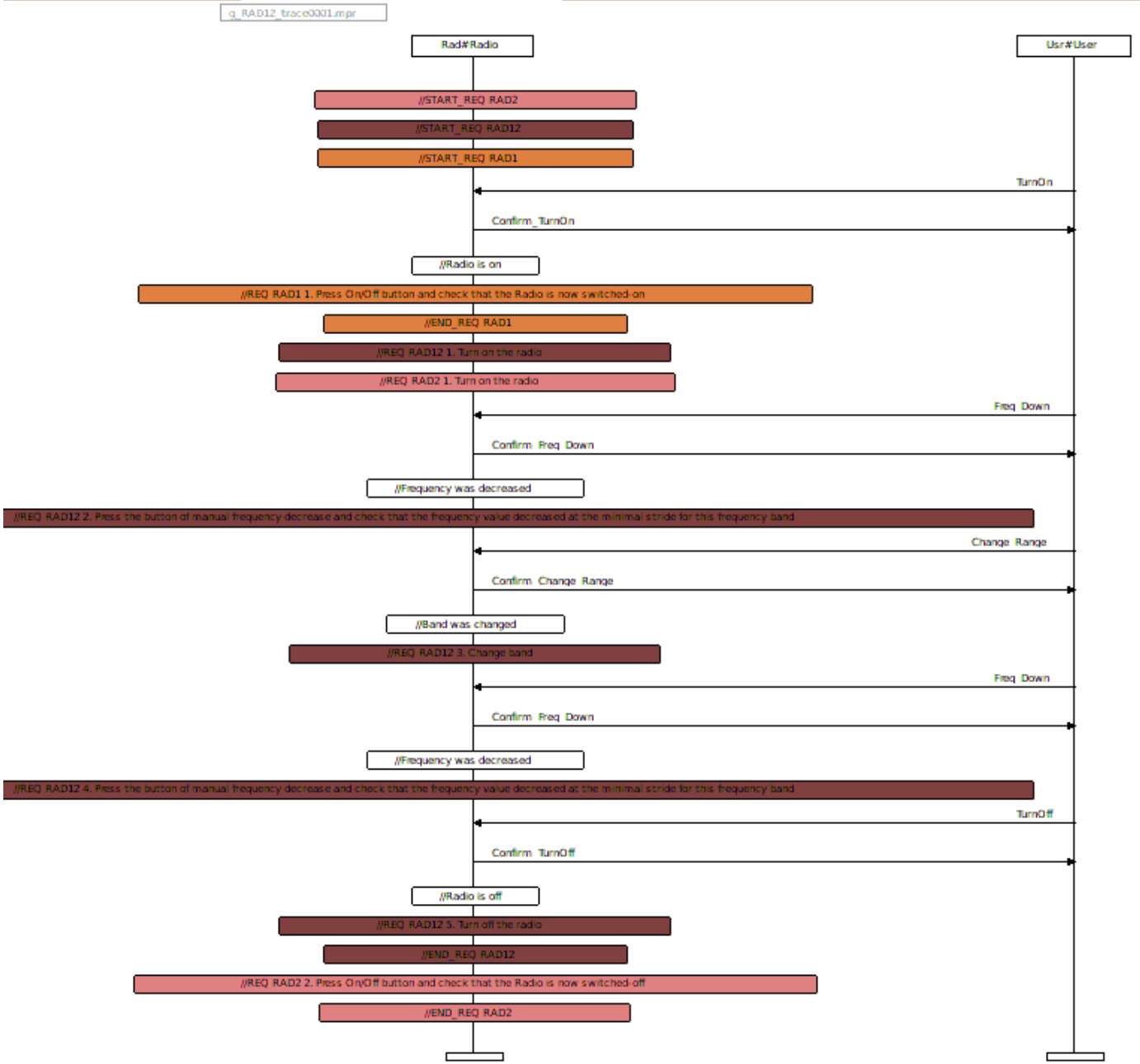
Поддерживается два режима конкретизации: автоматический и определенный пользователем.

Symbolic Parameter Type	Possible Range	User option (left, middle, right)	Value in Concretized Trace
integer	[1;9]	left	1
integer	[1;9]	middle	5
integer	[1;9]	right	9
enumerated	val1,val2,val3,val4	left	val1
enumerated	val1,val2,val3,val4	middle	val2
enumerated	val1,val2,val3,val4	right	val4

Трасса, доказанная верификатором



Тест, полученный из трассы



Журнал прогона тестов

Analysis report

Project "ProjName"

Total tests: 18
Passed: 94%
17

- Test g_RAD25_2_trace0001 Passed
- Test g_RAD5_trace0001 Passed
- Test g_RAD23_1_trace0001 Passed
- Test g_RAD6_trace0001 Passed
- Test g_RAD14_trace0001 Passed
- Test g_RAD9_trace0001 Passed
- Test g_RAD24_trace0001 Passed
- Test g_RAD7_trace0001 Passed
- Test g_RAD18_trace0001 Passed
- Test g_RAD8_trace0001 Passed
- Test g_RAD21_trace0001 Passed
- Test g_RAD16_trace0001 Passed
- Test g_RAD23_2_trace0001 Passed
- Test g_RAD25_1_trace0001 Passed
- Test g_RAD12_trace0001 Passed
- Test g_RAD20_trace0001 Passed
- Test g_RAD21_trace0001 Passed
- Test g_RAD21_trace0001 Failed

Детальный Log теста

Task g_RAD25_2_trace0001 report

Project "ProjName"

Master sample:	/data/nvv/eclipse_infotek/CarRadio_JAVA/TAT/tmp/macroprc/g_RAD2
Test outcome:	/data/nvv/eclipse_infotek/CarRadio_JAVA/TAT/log/mprlog/g_RAD25_2
Comments:	Traces are equal.

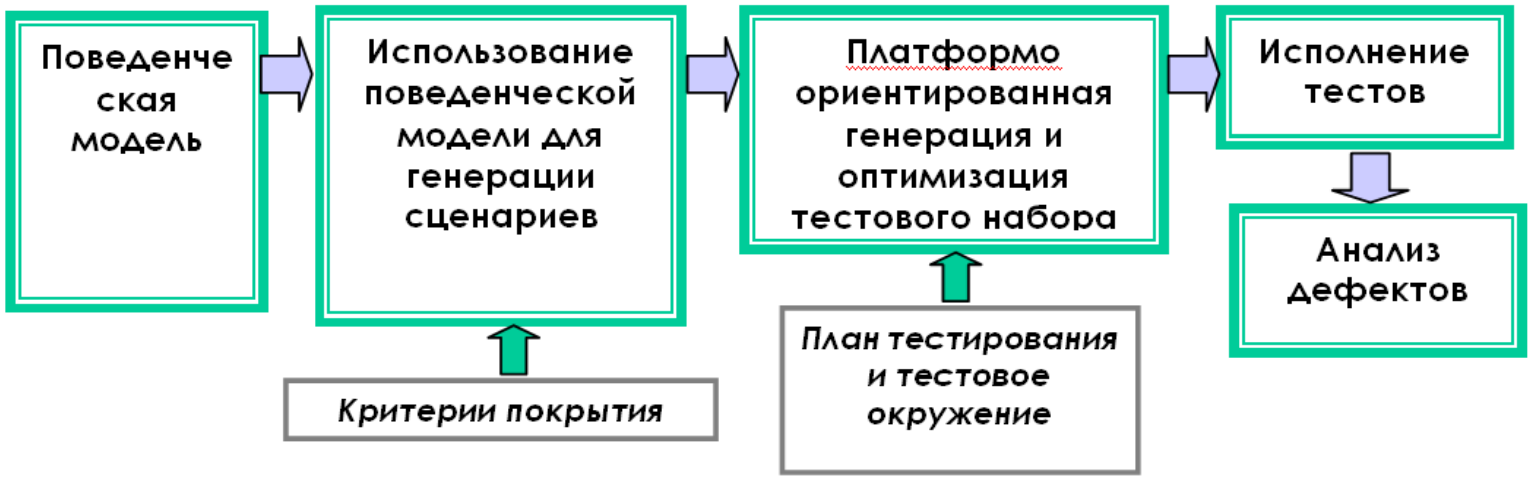
Sample

```
1: mscdocument g_RAD25_2_trace0001N0;
2: msc g_RAD25_2_trace0001N0/*["/data/nvv/eclipse_demos/CarRadio_JAVA/TAT/mpr/g_RAD25_2_trace0001.mpr",1,5,31]*/;
3: SUT: instance;
4: TAT: instance;
5: all: condition /*{0}*/TAT_SYNCHRONIZATION;
6: SUT: action 'START_REQ RAD25_2'/*[0,5]*/;
7: all: condition /*{0}*/TAT_SYNCHRONIZATION;
8: TAT: out TurnOn,1 to SUT via Rad_Usr/*[0,6]*/;
9: SUT: in TurnOn,1 from TAT/*[0,7]*/;
10: all: condition /*{0}*/TAT_SYNCHRONIZATION;
11: SUT: out Confirm_TurnOn,2 to TAT/*[0,8]*/;
12: TAT: in Confirm_TurnOn,2 from SUT/*[0,9]*/;
13: all: condition /*{0}*/TAT_SYNCHRONIZATION;
14: SUT: action '//Radio is on'/*[0,10]*/;
15: all: condition /*{0}*/TAT_SYNCHRONIZATION;
16: SUT: action 'REQ RAD25_2 1. Turn on the radio'/*[0,11]*/;
17: all: condition /*{0}*/TAT_SYNCHRONIZATION;
18: TAT: out Memory,3 to SUT via Rad_Usr/*[0,12]*/;
19: SUT: in Memory,3 from TAT/*[0,13]*/;
20: all: condition /*{0}*/TAT_SYNCHRONIZATION;
21: SUT: out Confirm_Memory,4 to TAT/*[0,14]*/;
22: TAT: in Confirm_Memory,4 from SUT/*[0,15]*/;
23: all: condition /*{0}*/TAT_SYNCHRONIZATION;
24: SUT: action '//Memory mode is on'/*[0,16]*/;
25: all: condition /*{0}*/TAT_SYNCHRONIZATION;
26: SUT: action 'REQ RAD25_2 2. Press the Memory button and check that the radio is in the memory mode'/*[0,17]*/;
27: all: condition /*{0}*/TAT_SYNCHRONIZATION;
28: TAT: out AutoSearch_Down,5 to SUT via Rad_Usr/*[0,18]*/;
29: SUT: in AutoSearch_Down,5 from TAT/*[0,19]*/;
30: all: condition /*{0}*/TAT_SYNCHRONIZATION;
31: SUT: action '//Autosearch was started'/*[0,20]*/;
32: all: condition /*{0}*/TAT_SYNCHRONIZATION;
33: SUT: action '//Autosearch was finished'/*[0,21]*/;
34: all: condition /*{0}*/TAT_SYNCHRONIZATION;
35: SUT: out Confirm_AutoSearch_Down,6 to TAT/*[0,22]*/;
36: TAT: in Confirm_AutoSearch_Down,6 from SUT/*[0,23]*/;
37: all: condition /*{0}*/TAT_SYNCHRONIZATION;
```

Outcome

```
1: mscdocument g_RAD25_2_trace0001N0;
2: msc g_RAD25_2_trace0001N0;
3: TAT: instance;
4: SUT: instance;
5: SUT: action 'START_REQ RAD25_2 ';
6: TAT: out TurnOn,1 to SUT; /* TIME:1000 */
7: SUT: in TurnOn,1 from TAT; /* TIME:1000 */
8: SUT: out Confirm_TurnOn,2 to TAT; /* TIME:1000 */
9: TAT: in Confirm_TurnOn,2 from SUT; /* TIME:1000 *
10: SUT: action '//Radio is on ';
11: SUT: action 'REQ RAD25_2 1. Turn on the radio ';
12: TAT: out Memory,3 to SUT; /* TIME:2001 */
13: SUT: in Memory,3 from TAT; /* TIME:2001 */
14: SUT: out Confirm_Memory,4 to TAT; /* TIME:2001 *
15: TAT: in Confirm_Memory,4 from SUT; /* TIME:2001
16: SUT: action '//Memory mode is on ';
17: SUT: action 'REQ RAD25_2 2. Press the Memory bu
18: TAT: out AutoSearch_Down,5 to SUT; /* TIME:3002
19: SUT: in AutoSearch_Down,5 from TAT; /* TIME:300
20: SUT: action '//Autosearch was started ';
21: SUT: action '//Autosearch was finished ';
22: SUT: out Confirm_AutoSearch_Down,6 to TAT; /* TI
23: TAT: in Confirm_AutoSearch_Down,6 from SUT; /* 7
24: SUT: action 'REQ RAD25_2 3. Press the button of a
```

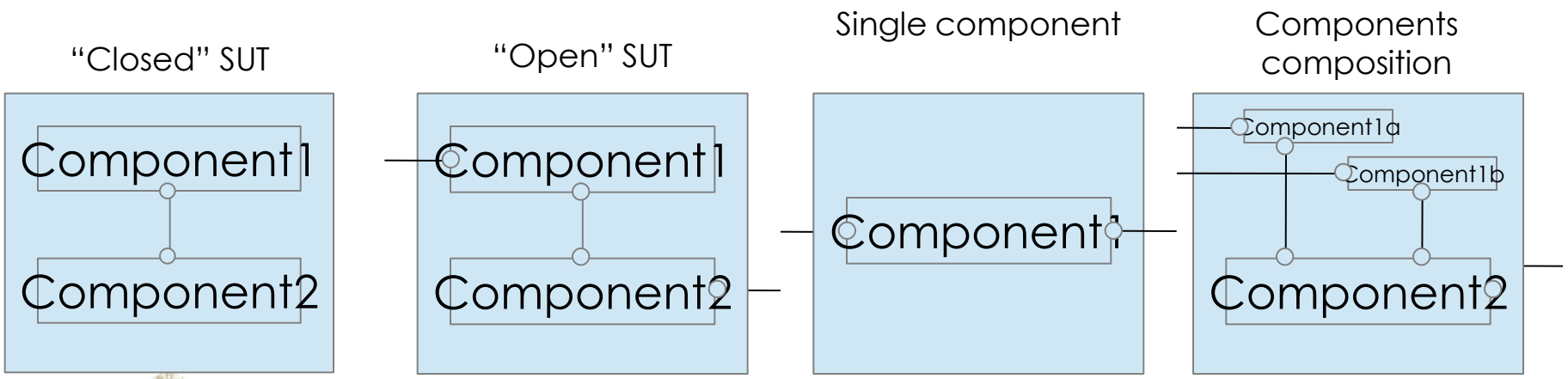
Свойства систем автоматизации тестирования



1)Автоматическое исполнение (прогон) тестового набора в соответствии с планом тестирования и автоматизированный анализ причин или мест ошибок.
2)Автоматическая генерация оптимизированного кода тестовых наборов (test-suite) на целевую платформу по поведенческим сценариям (трассам).
3)Использование поведенческой модели для автоматической генерации сценариев, удовлетворяющих заданному критерию покрытия.
4)Автоматическая генерация поведенческой модели программного продукта на основе, созданных вручную формальных моделей отдельных требований

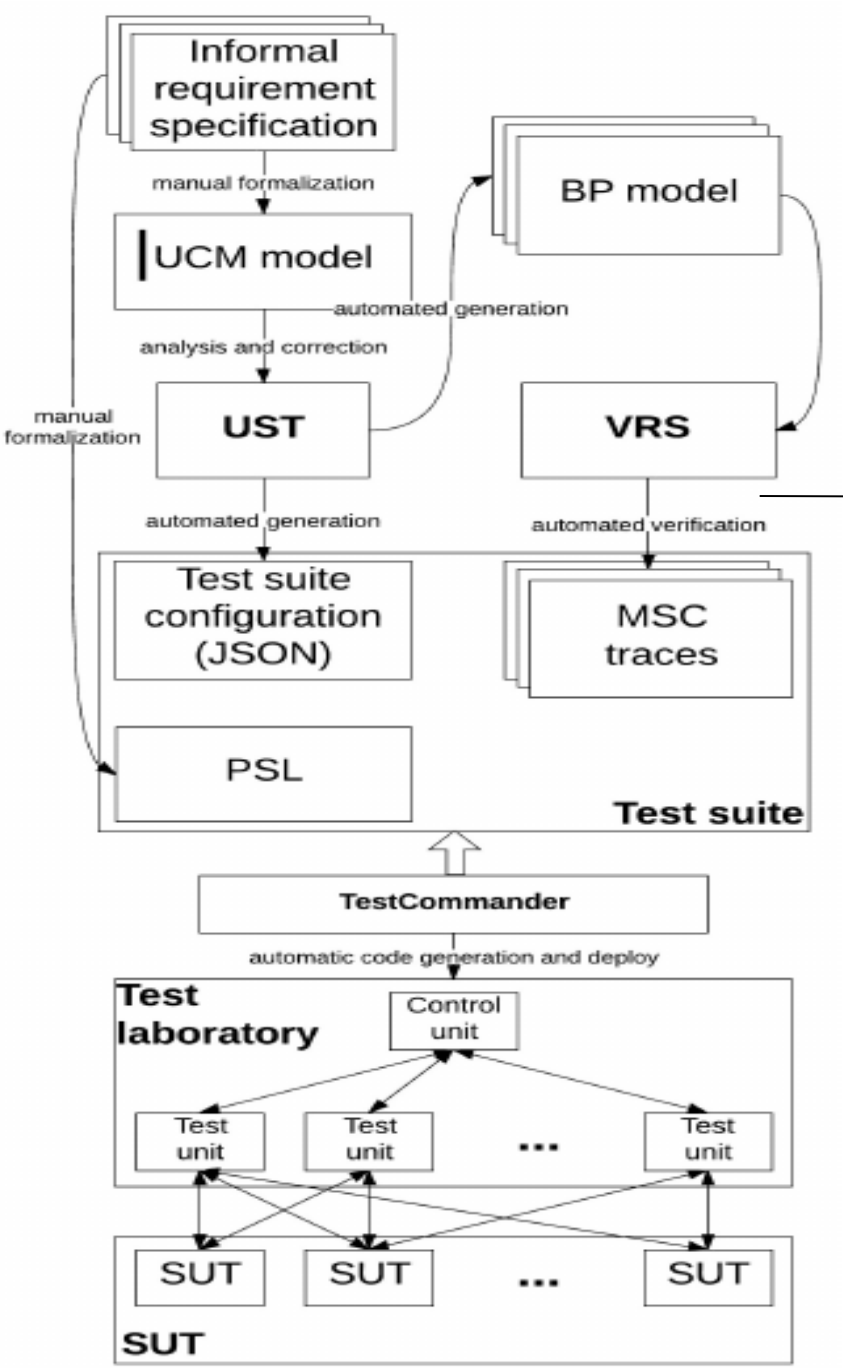
TAT: Методы тестирования

- **Black-box тестирование открытых систем (с внешними входами и выходами)**
- **Системное тестирование открытых систем**
- **Тестирование отдельных компонент целых систем**
- **Тестирование произвольных комбинаций компонент распределенных систем**



Интеграция верификации и тестирования

- **Генерируемый тестовый набор состоит из одного или нескольких тестирующих и одного управляющего модулей**
- **Генерируемый конфигурационный файл позволяет определить физическое расположение модулей тестового набора и SUT.**
- **Набор инструментов автоматизации тестирования обеспечивает свойства масштабируемости и адаптируемости под условия конкретной задачи**



ЗАКЛЮЧЕНИЕ

- **Реализация концепции осуществлена в интегрированной инструментальной среде VRS/TAT, что позволяет обеспечить эффективное применение инструментария и поддерживающей технологии на проектах средней и большой сложности из области телекоммуникационных приложений.**
- **Следует отметить, что наряду с внедрением перспективных методов, обеспечивающих совершенствование производства программного продукта, актуальной остается проблема их баланса в практике применения в промышленных программных проектов среднего и большого размера.**

Большое спасибо