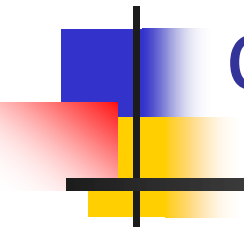


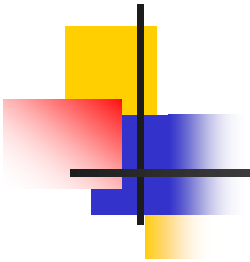
Лекция 8. Тестирование на основе формальных моделей

А.К.Петренко, А.В.Хорошилов,
Е.В.Корныхин

МГУ ВМК, ИСП РАН

http://sp.cmc.msu.ru/courses/fmsp

Осень, 2012



План

- Постановка задачи Formal testing
- Model Based Testing (MBT)

Вопрос. Что делать, если строго доказать не удается?

Ответ -

Разработать конечный набор тестов и при этом:

- Строго сформулировать предположения (гипотезы) – какие мы делаем допущения, полагаясь на протестированную программу.
- Строго сформулировать критерии адекватности/полноты набора тестов.
- Строго сформулировать критерии корректности
- Строго сформулировать критерии соответствия между моделями и реализацией

О классах эквивалентности

- Тестовые ситуации, относящиеся к одному классу эквивалентности, имеют одинаковые возможности в плане выявления нарушения критериев корректности

Это одновременно, суть тестирования и гипотеза, на которой базируются наши ожидания о результативности тестирования

Критерии адекватности/полноты набора тестов

```
is_triangle: Real >< Real >< Real -~-> Kind
is_triangle(x, y, z) as kind
  post
  if x=y /\ x=z then kind=equilateral
  elsif x=y \/ x=z \/ y=z
                then kind=isosceles
  else kind=common
  end
  pre      x+y ≥ z /\ x+z ≥ y /\ y+z ≥ x /\
           x>0 /\ y>0 /\ z>0
```

Критерии адекватности/полноты набора тестов. Короткая или классическая логика

```
is_triangle: Real >< Real >< Real -~-> Kind
is_triangle(x, y, z) as kind
   post let a=(x=y),b=(x=z),c=(x=y),d=(x=z),e=(y=z) in
   if a /\ b then kind=equilateral
   elsif c \/ d \/ e then kind=isosceles
   else kind=common
   end end
   pre x+y ≥ z /\ x+z ≥ y /\ y+z ≥ x /\ x>0 /\ y>0 /\ z>0
```

Классы эквивалентности:

ab; ~ac; ~a~cd; ~a~c~de; ~a~c~d~e; a~bc; a~b~cd; a~b~c~de; a~b~c~d~e

Вопрос: Почему мы не выделили ситуацию ~ab~cd?

Для ветви elsif. Случай короткой логики:
6 классов – ~ac, ~a~cd, ~a~c~de, a~bc, a~b~cd, a~b~c~de (ДНФ)
Случай классической логики:
21 класс – ~abcde, a~bcde, ~a~bcde, ~ab~cde,a~b~cde,~a~b~cde, … (СДНФ)

Когда короткая логика существенна?

if (x ∈ **dom** m) /\ m(x)=y **then true else … end**
или

let a=(x ∈ **dom** m) , b=(m(x)=y) **in**
if a /\ b **then true else … end end**

Здесь классы эквивалентности
ab, a~b, ~a
Случаи ~ab и ~a~b невычислимы.

Критерии адекватности/полноты набора тестов. Классическая логика

```
is_triangle: Real >< Real >< Real -~-> Kind
is_triangle(x, y, z) as kind
   post let a=(x=y),b=(x=z),c=(x=y),d=(x=z),e=(y=z) in
   if a /\ b then kind=equilateral
   elsif c \/ d \/ e then kind=isosceles
   else kind=common
   end end
   pre x+y ≥ z /\ x+z ≥ y /\ y+z ≥ x /\ x>0 /\ y>0 /\ z>0
```

Классы эквивалентности:

ab,
~abcde,~ab~cde,~abc~de, ~abcd~e, ~ab~c~de, ~ab~cd~e, ~abc~d~e, ~ab~c~d~e,
a~bcde,a~b~cde,a~bc~de, a~bcd~e, a~b~c~de, a~b~cd~e, a~bc~d~e, a~b~c~d~e,
~a~bcde,~a~b~cde,a~bc~de,~a~bcd~e,~a~b~c~de,~a~b~cd~e,~a~bc~d~e,~a~b~c~d~e

Используется классическая логика для анализа условных выражений

If-выражения задают короткую логику.

Критерии адекватности/полноты набора тестов. Исключение тождественно ложных термов

```
is_triangle: Real >< Real >< Real -~-> Kind
is_triangle(x, y, z) as kind
   post let a=(x=y),b=(x=z),c=(x=y),d=(x=z),e=(y=z) in
   if a /\ b then kind=equilateral
   elsif c \/ d \/ e then kind=isosceles
   else kind=common
   end end
   pre x+y ≥ z /\ x+z ≥ y /\ y+z ≥ x /\ x > 0 /\ y > 0
   /\ z > 0
```

Вопрос: Имеет ли смысл рассматривать ситуацию ab~cde?

То есть мы пытаемся определить класс, где

(x=y) (x=z) ~ (x=y) (x=y) (y=z)

Исключение тождественно ложных термов (1)

```
is_triangle: Real >< Real >< Real -~-> Kind
is_triangle(x, y, z) as kind
   post let a=(x=y),b=(x=z),c=(x=y),d=(x=z),e=(y=z) in
   if a /\ b then kind=equilateral
   elsif c \/ d \/ e then kind=isosceles
   else kind=common
   end end
   pre x+y ≥ z /\ x+z ≥ y /\ y+z ≥ x /\ x>0 /\ y>0 /\ z>0
```

Классы эквивалентности:

ab;

~abcde,~ab~cde,~abc~de, ~abcd~e, ~ab~c~de, ~ab~cd~e, ~abc~d~e, ~ab~c~d~e,

a~bcde,a~b~cde,a~bc~de, a~bcd~e, a~b~c~de, a~b~cd~e, a~bc~d~e, a~b~c~d~e,

~a~bcde,~a~b~cde,~a~bc~de,~a~bcd~e,~a~b~c~de,~a~b~cd~e,~a~bc~d~e,~a~b~c~d~e

Исключение тождественно ложных термов (2)

```
is_triangle: Real >< Real >< Real -~-> Kind
is_triangle(x, y, z) as kind
   post let a=(x=y),b=(x=z),c=(x=y),d=(x=z),e=(y=z) in
   if a /\ b then kind=equilateral
   elsif c \/ d \/ e then kind=isosceles
   else kind=common
   end end
   pre x+y ≥ z /\ x+z ≥ y /\ y+z ≥ x /\ x>0 /\ y>0 /\ z>0
```

Классы эквивалентности:

ab;

~abcde,~ab~cde,~abc~de, ~abcd~e, ~ab~c~de, ~ab~cd~e, ~abc~d~e, ~ab~c~d~e,
a~bcde,a~b~cde,a~bc~de, a~bcd~e, a~b~c~de, a~b~cd~e, a~bc~d~e, a~b~c~d~e,
~a~bcde,~a~b~cde,~a~bc~de,~a~bcd~e,~a~b~c~de,~a~b~cd~e,~a~bc~d~e,~a~b~c~d~e

Исключение тождественно ложных термов (3)

```
is_triangle: Real >< Real >< Real -~-> Kind
is_triangle(x, y, z) as kind
   post let a=(x=y),b=(x=z),c=(x=y),d=(x=z),e=(y=z) in
   if a /\ b then kind=equilateral
   elsif c \/ d \/ e then kind=isosceles
   else kind=common
   end end
   pre x+y ≥ z /\ x+z ≥ y /\ y+z ≥ x /\ x>0 /\ y>0 /\ z>0
```

Классы эквивалентности:

ab;

~ababe,~ab~abe,~aba~be,~abab~e,~ab~a~be, ~ab~ab~e, ~aba~b~e, ~ab~a~b~e,
a~babe,a~b~abe,a~ba~be,a~bab~e,a~b~a~be, a~b~ab~e, a~ba~b~e, a~b~a~b~e,
~a~babe,~a~b~abe,~a~ba~be,~a~bab~e,~a~b~a~be,~a~b~ab~e,~a~ba~b~e,~a~b~a~b~e

Исключение тождественно ложных термов (4)

```
is_triangle: Real >< Real >< Real -~-> Kind
is_triangle(x, y, z) as kind
   post let a=(x=y),b=(x=z),c=(x=y),d=(x=z),e=(y=z) in
   if a /\ b then kind=equilateral
   elsif c \/ d \/ e then kind=isosceles
   else kind=common
   end end
   pre x+y ≥ z /\ x+z ≥ y /\ y+z ≥ x /\ x>0 /\ y>0 /\ z>0
```

Классы эквивалентности:

ab;

~ababe,~ab~abe,~aba~be,~abab~e, ~ab~a~be, ~ab~ab~e, ~aba~b~e, ~ab~a~b~e,
a~babe,a~b~abe,a~ba~be,a~bab~e, a~b~a~be, a~b~ab~e, a~ba~b~e, a~b~a~b~e,
~a~babe,~a~b~abe,~a~ba~be,~a~bab~e,~a~b~a~be,~a~b~ab~e,~a~ba~b~e,~a~b~a~b~e

Исключение тождественно ложных термов (5)

```
is_triangle: Real >< Real >< Real -~-> Kind
is_triangle(x, y, z) as kind
   post let a=(x=y),b=(x=z),c=(x=y),d=(x=z),e=(y=z) in
   if a /\ b then kind=equilateral
   elsif c \/ d \/ e then kind=isosceles
   else kind=common
   end end
   pre x+y ≥ z /\ x+z ≥ y /\ y+z ≥ x /\ x>0 /\ y>0 /\ z>0
```

Классы эквивалентности:

ab;

~abe, , , , ~ab~e, , ,
, ,a~be, , , , a~b~e,
,
, , , ,~a~be, ,
,~a~b~e,

Исключение тождественно ложных термов (6). Обсуждение

```
is_triangle: Real >< Real >< Real -~-> Kind
is_triangle(x, y, z) as kind
   post let a=(x=y),b=(x=z),c=(x=y),d=(x=z),e=(y=z) in
   if a /\ b then kind=equilateral
   elsif c \/ d \/ e then kind=isosceles
   else kind=common
   end end
   pre x+y ≥ z /\ x+z ≥ y /\ y+z ≥ x /\ x>0 /\ y>0 /\
   z>0
```

Классы эквивалентности в соответствии с классической логикой:

ab;
~abe, ~ab~e,
a~be, a~b~e,
~a~be, ~a~b~e

Классы эквивалентности в соответствии с короткой логикой:

ab; ~ab; ~a~be; ~a~b~e; a~b;

А все ли зависимости учтены?

Спецификация на случай сравнений длин сторон с ограниченной точностью

```
value eps : Real,
is_triangle: Real >< Real >< Real -~-> Kind
is_triangle(x, y, z) as kind
   post let a=(abs(x-y)<eps),b=(abs(x-z)<eps),e=(abs(y-z)<eps) in
   if a /\ b /\ e then kind=equilateral
   elsif a \/ b \/ e then kind=isosceles
   else kind=common
   end end
   pre x+y ≥ z /\ x+z ≥ y /\ y+z ≥ x /\ x>0 /\ y>0 /\ z>0
```

Классы эквивалентности в соответствии с классической логикой:

abe;
~abe, ~ab~e,
a~be, a~b~e,
~a~be, ~a~b~e

Классы эквивалентности в соответствии с короткой логикой:

ab; ~ab; ~a~be; ~a~b~e; a~b

Классы эквивалентности входных данных. Тестовые наборы

Пусть: eps=0,5,
a=(abs(x-y)<eps), b=(abs(x-z)<eps), e=(abs(y-z)<eps)

Короткая логика	Классическая логика	Длины сторон (x; y; z)	Вид треугольника
abe	abe	1,0; 1,0; 1,0	Равносторонний
ab~e	ab~e	1,3; 1,6; 1,0	Равнобедренный
~ab	~ab~e	1,3; 1,9; 1,0	Равнобедренный
~ab	~abe	1,3; 1,9; 1,6	Равнобедренный
a~b	a~b~e	1,3; 1,0; 1,9	Равнобедренный
a~b	a~be	1,3; 1,6; 1,9	Равнобедренный
~a~be	~a~be	1,2; 2,0; 2,0	Равнобедренный
~a~b~e	~a~b~e	1,0; 2,0; 2,5	Общего вида

Пример реализации (с ошибкой)

```
value eps : Real,
is_triangle: Real >< Real >< Real -~-> Kind
is_triangle(x, y, z) is
   let a=(abs(x-y)<eps),b=(abs(x-z)<eps),e=(abs(y-z)<eps)
   in
        if a /\ b /\ e then equilateral
        elsif a/\~b/\~e \/ ~a/\b/\~e \/ ~a/\~b/\e then
   isosceles else common
        end
   end
```

Автор этой реализации решил, что равнобедренный треугольник, это тот, в котором только одна пара сторон имеет равную с точностью до эпсилон длину, хотя в силу нашего определение это не так.

Результаты прогона тестов

Короткая логика	Классическая логика	Длины сторон (x; y; z)	Вид треугольника	Вердикт
abe	abe	1,0; 1,0; 1,0	Равносторонний	Passed
ab~e	ab~e	1,3; 1,6; 1,0	Равнобедренный	Passed
~ab	~ab~e	1,3; 1,9; 1,0	Равнобедренный	Passed
~ab	~abe	1,3; 1,9; 1,6	Равнобедренный	**Failure**
a~b	a~b~e	1,3; 1,0; 1,9	Равнобедренный	Passed
a~b	a~be	1,3; 1,6; 1,9	Равнобедренный	**Failure**
~a~be	~a~be	1,2; 2,0; 2,0	Равнобедренный	Passed
~a~b~e	~a~b~e	1,0; 2,0; 2,5	Общего вида	Passed

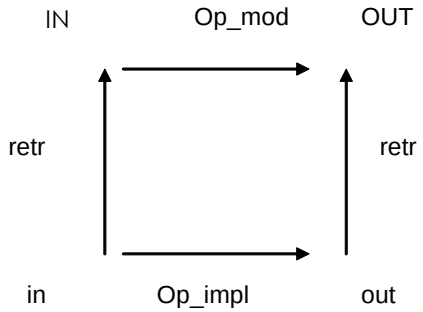
При использовании короткой логики ошибки могут быть пропущены.

Строго сформулировать критерии корректности

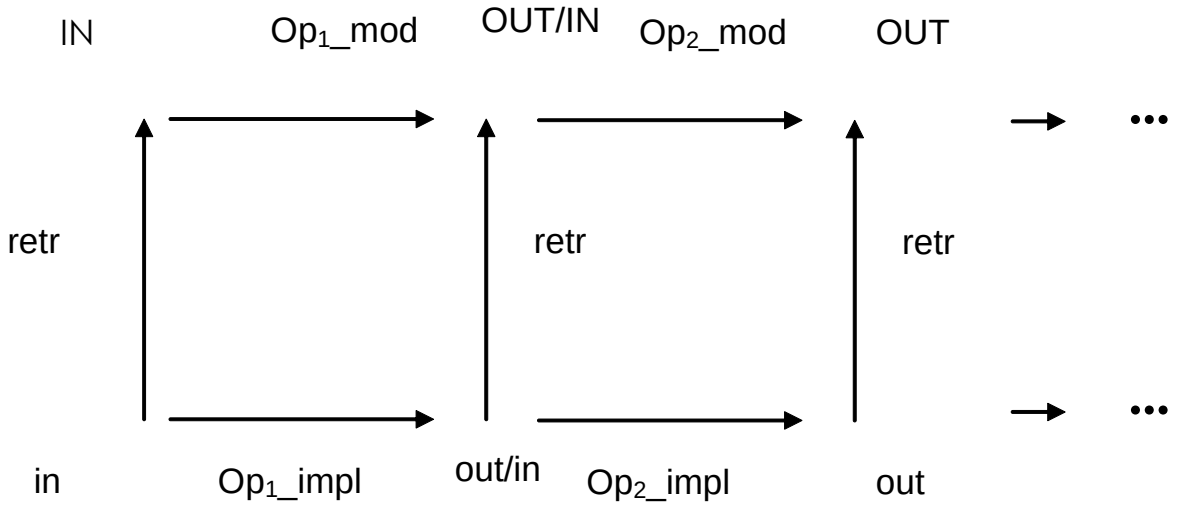
Построение тестового оракула (оракул выносит вердикт о выполнении критерия):

- При наличии имплицитной спецификации критерий - post-условие
- При наличии только явной спецификации требуется функция проверки эквивалентности результатов модели и реализации

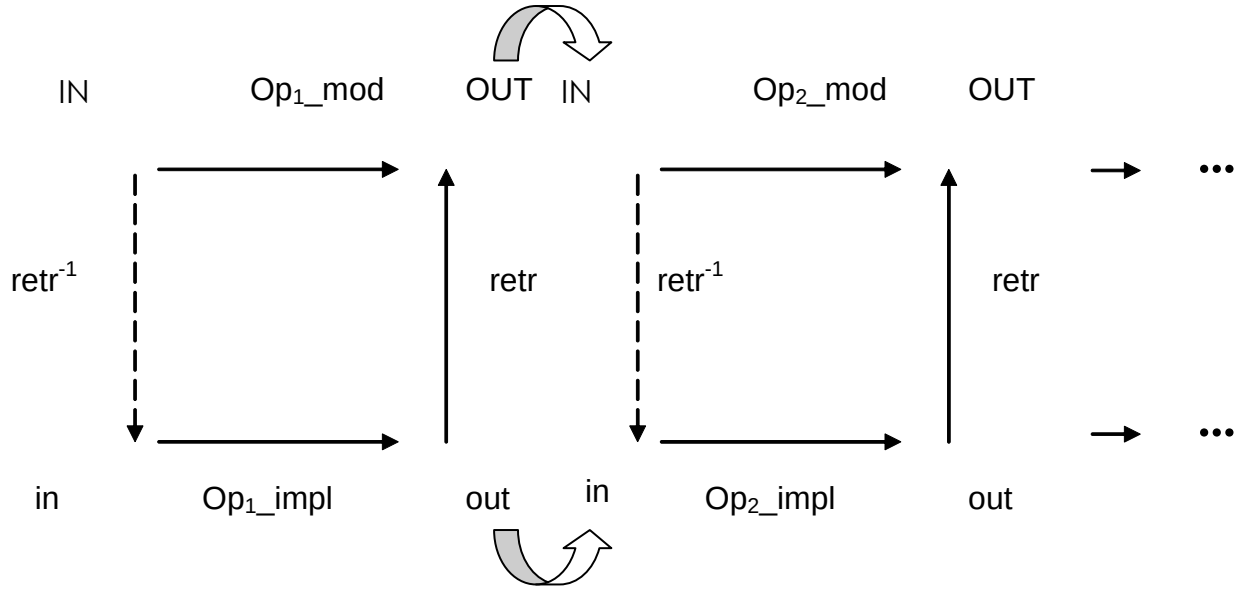
Строго сформулировать критерии соответствия между моделями и реализацией (1)



Случай цепочки операций

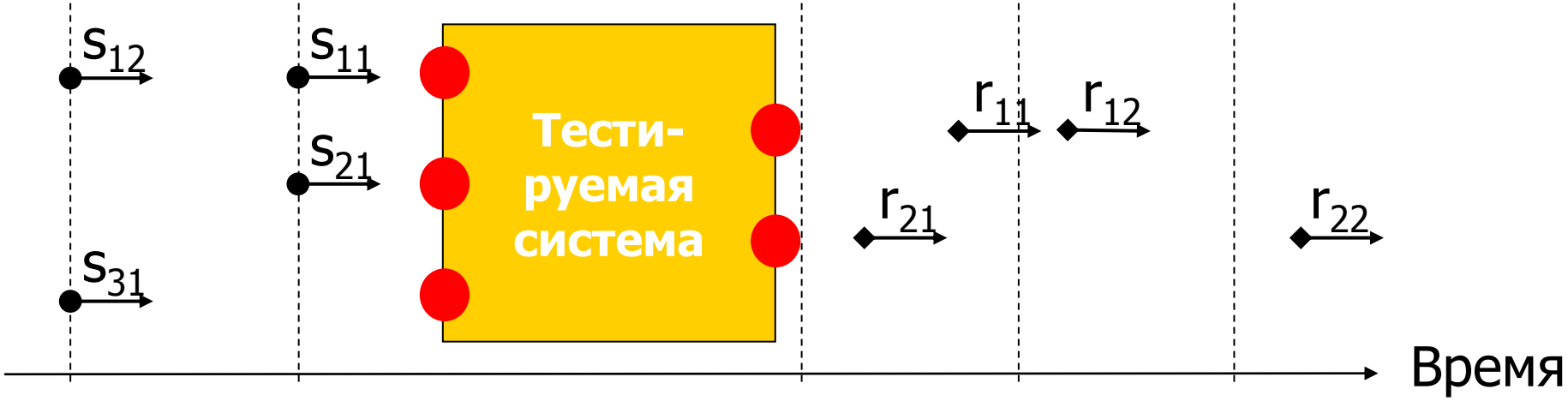


Строго сформулировать критерии соответствия между моделями и реализацией (2)



Как уклониться от реализации обратной retr функции

Открытые вопросы: случай параллельных процессов и асинхронных взаимодействий?



Имеется несколько последовательностей воздействий (стимулов) и реакций.

Воздействия и реакции - частично упорядоченное множество событий.

Вопрос – Каковы критерии корректности ?

Можно ли использовать метрики покрытия на основе ДНФ или СДНФ в коде реализации?

- Критерии C0, C1, C2
- MC/DC
- Открытые вопросы:
  - метрики тестирования OO и event-driven, etc.

Можно ли использовать метрики покрытия на основе ДНФ или СДНФ в коде реализации? (2)

Modified condition/decision coverage (MC/DC), is used in the standard DO-178B to ensure that Level A software is tested adequately.

- Each decision tries every possible outcome
- Each condition in a decision takes on every possible outcome
- Each entry and exit point is invoked
- Each condition in a decision is shown to independently affect the outcome of the decision.
- Independence of a condition is shown by proving that only one condition changes at a time.

Модифицированный метод покрытия по веткам/условиям (Modified Condition/Decision Coverage или MC/DC)

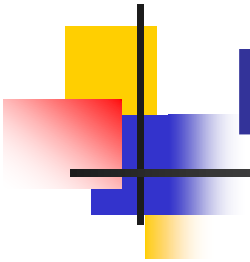
Для обеспечения полного покрытия по этому методу необходимо выполнение следующих условий:

- каждое логическое условие должно принимать все возможные значения;
- каждая компонента логического условия должна хотя бы один раз принимать все возможные значения;
- должно быть показано независимое влияние каждой из компонент на значение логического условия, т. е. влияние при фиксированных значениях остальных компонент.

(Н.Налютин http://software-testing.ru/files/testing_magazine.pdf)

Как MBT (Model Based Testing) выглядит на практике?

- См.:
  - Rational Rhapsody (IBM), SCADE (Esterel Technology)
  - SpecExplorer (Microsoft Research)
  - Nmodel (C#); TestNG, Summer (Java)
  - UniTESK

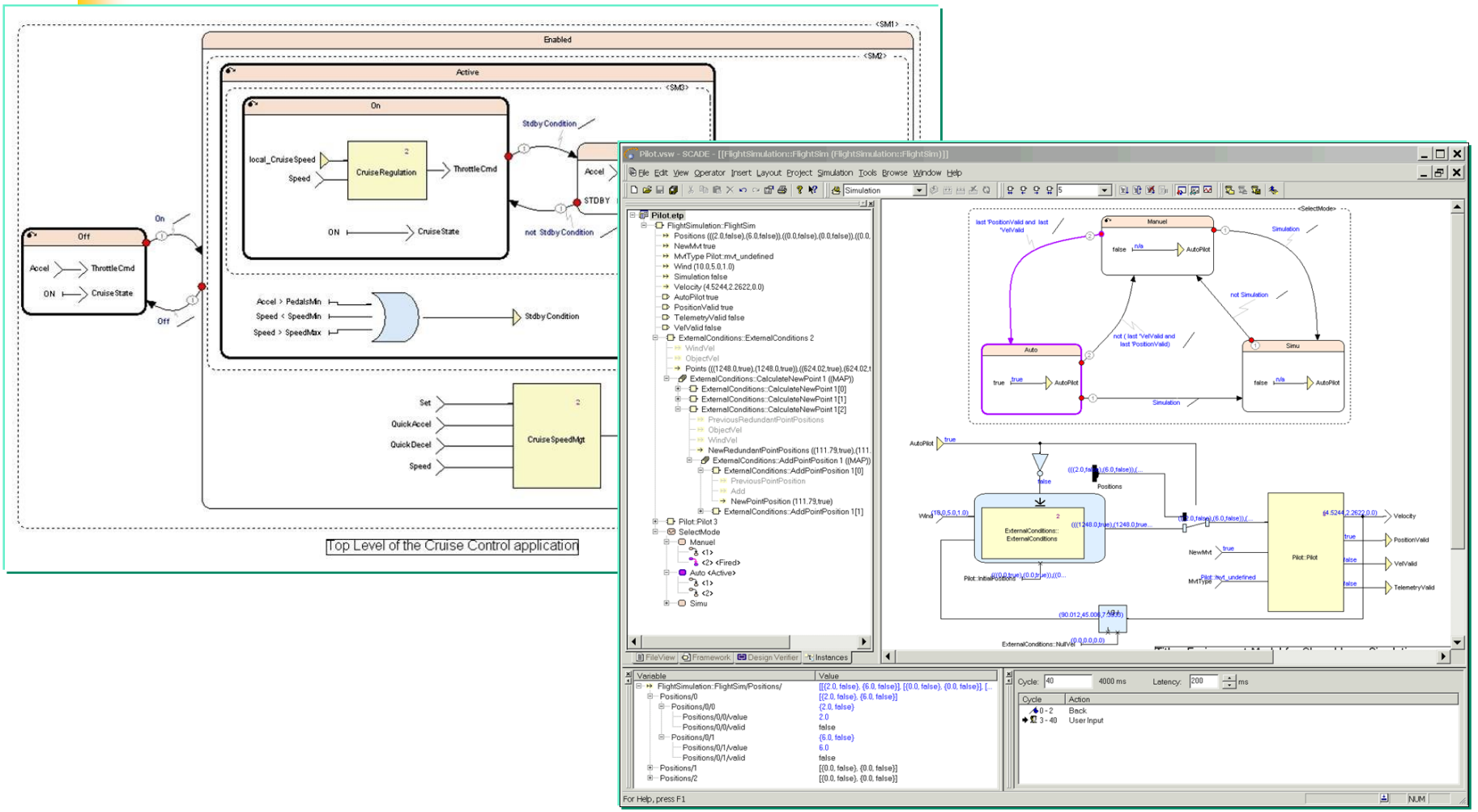


Formal Testing

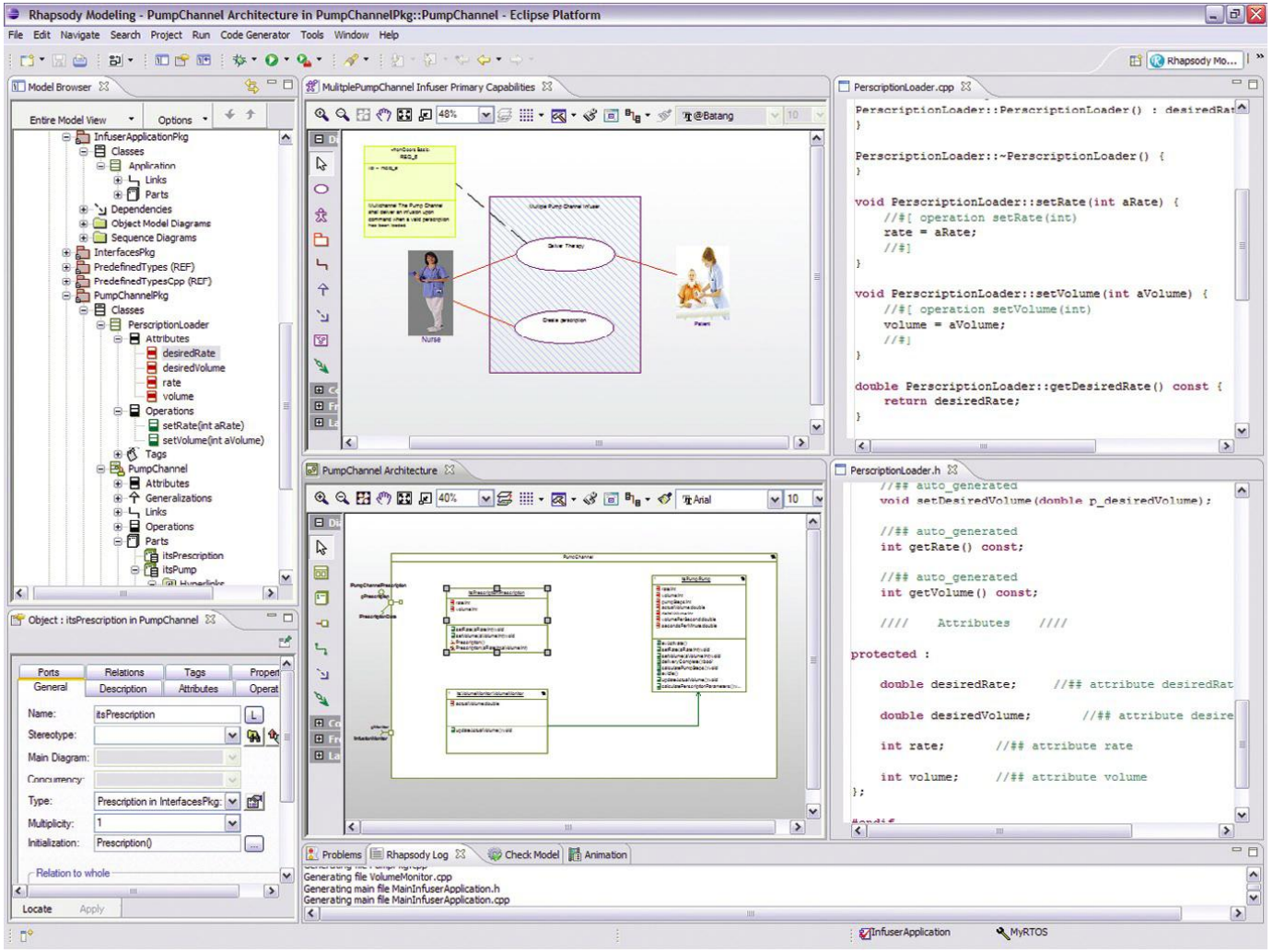
- *Acceptance test*: a formal test defined to check acceptance criteria for a software (Wikipedia)
- Formal testing: Testing based on formal specifications, models, and formal methods

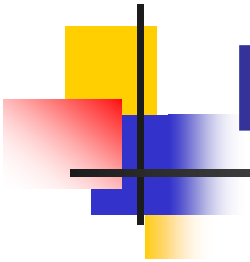
- Боб Биндер – Any testing is model based testing!

Модели и тестирование на их основе. SCADE (Esterel Technologies)



Модели и тестирование на их основе. Rhapsody (IBM)

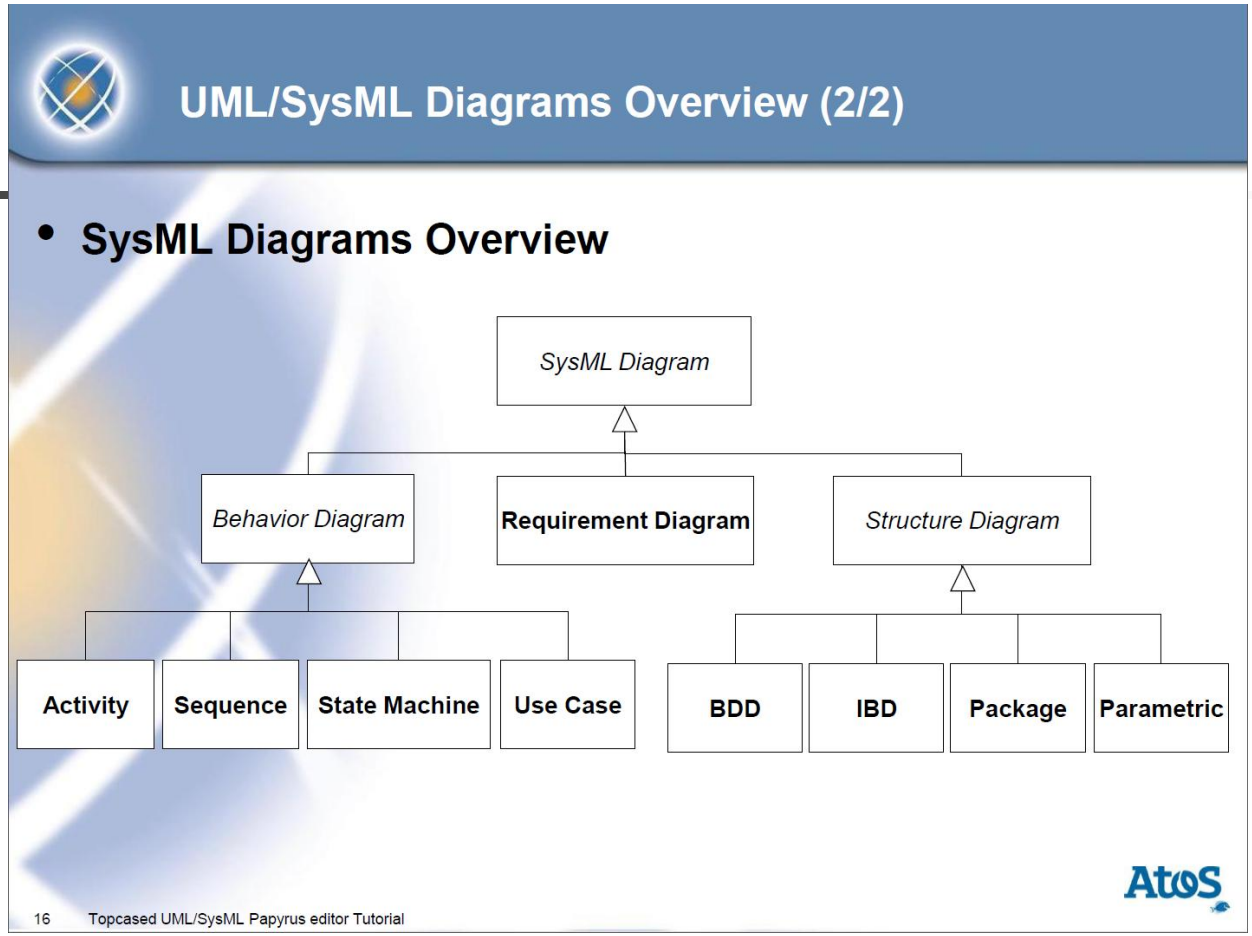




Новые языки моделирования

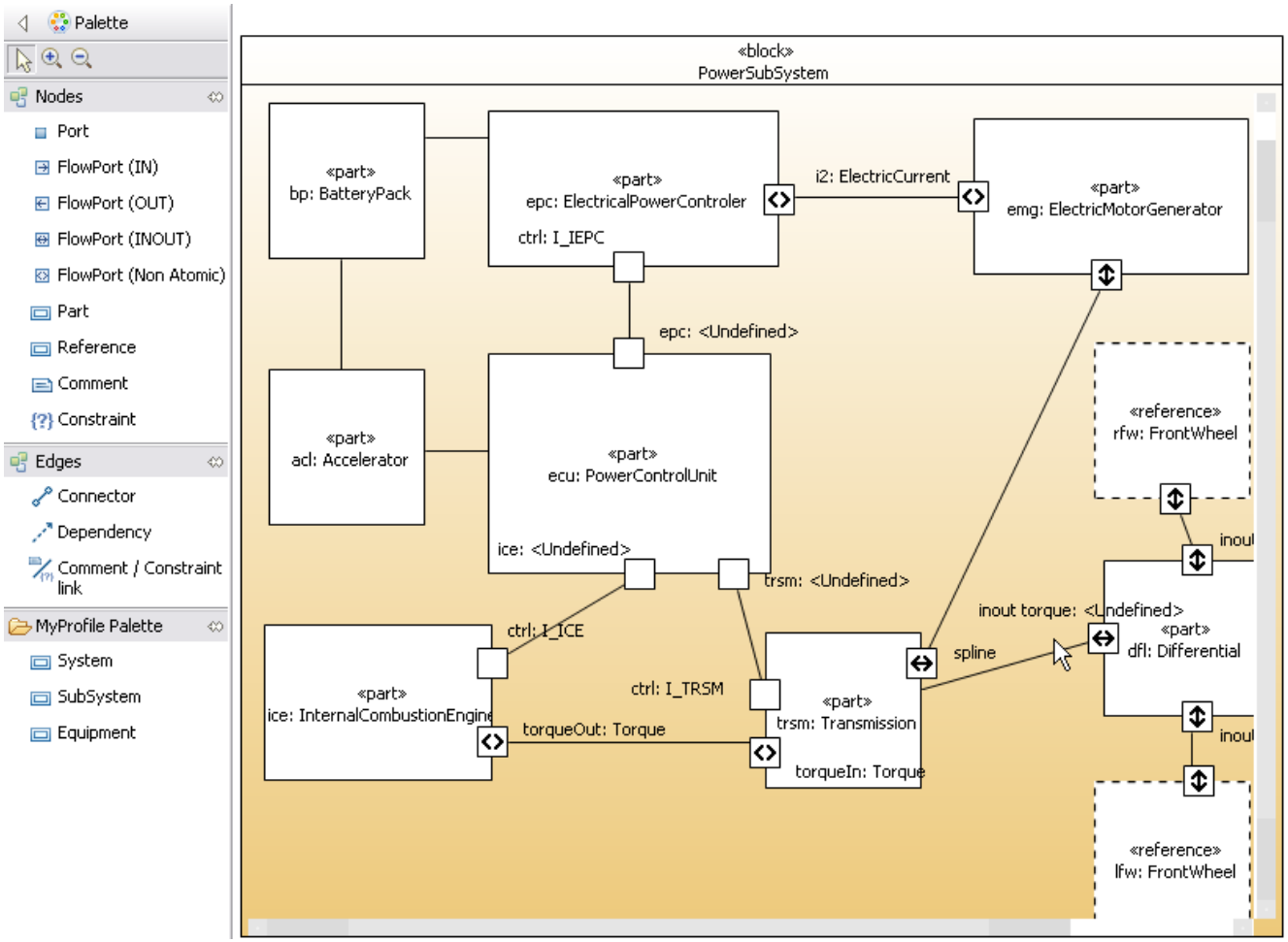
- SysML – UML профиль
- AADL –

  Architecture Analysis & Design Language

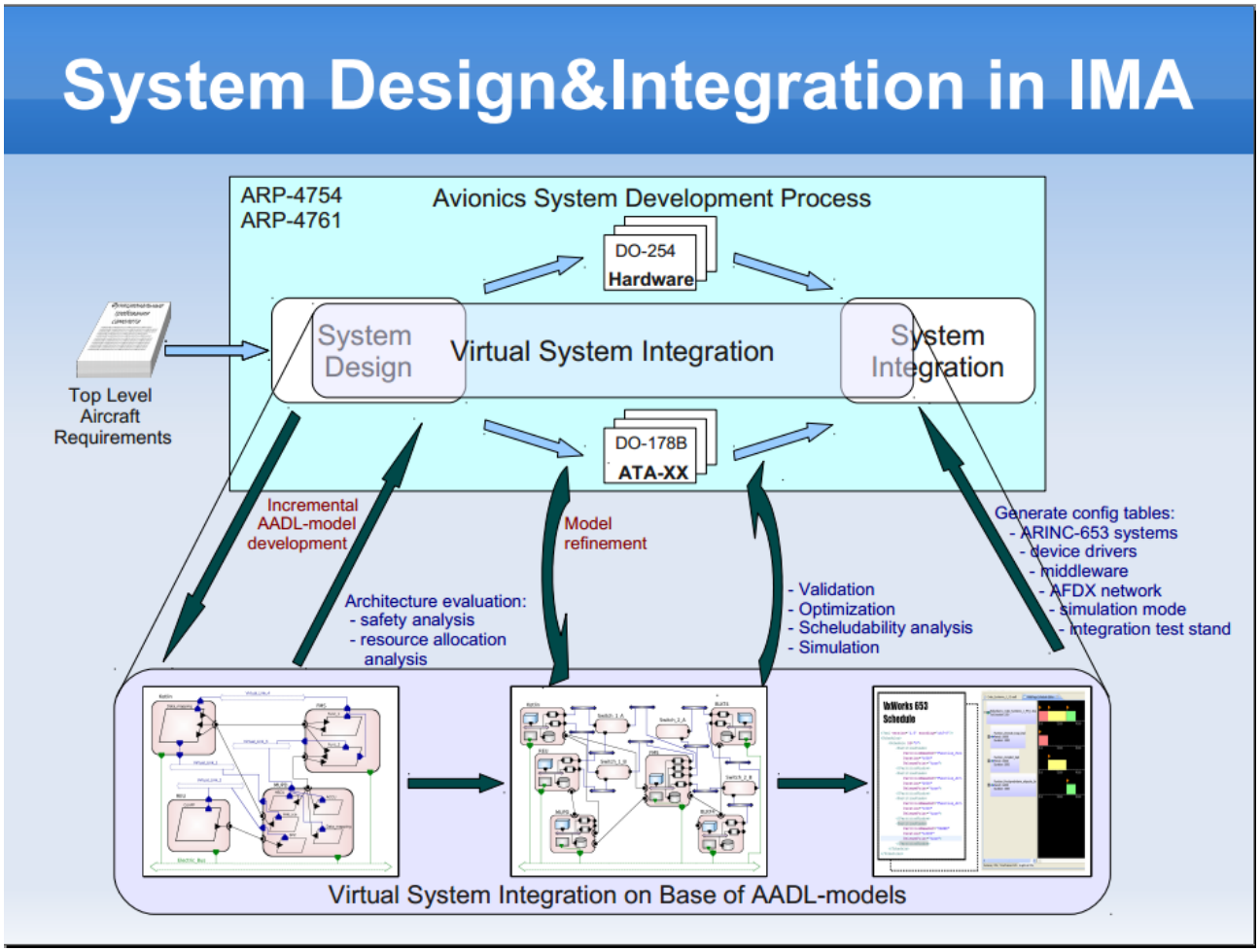


- Block Definition Diagram (BDD)
- Internal Block Diagram (IBD)
- Parametric Diagram

SysML Internal Block Diagram (IBD)

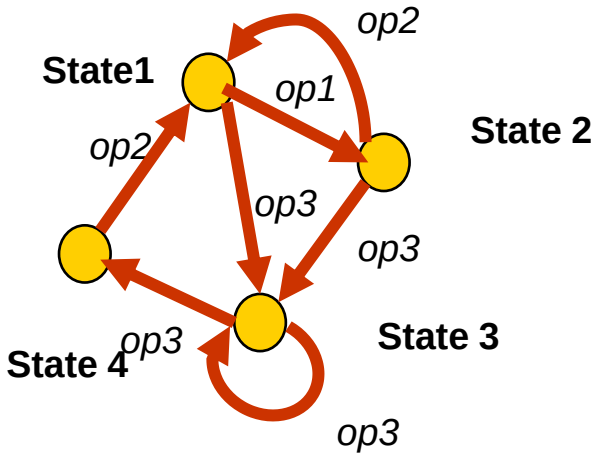


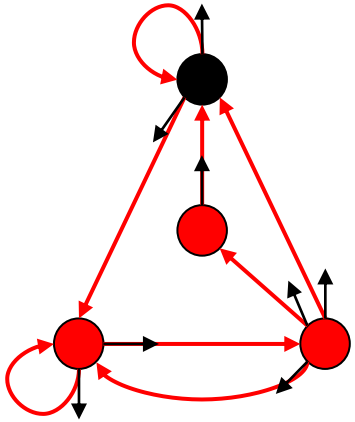
Пример моделей на AADL

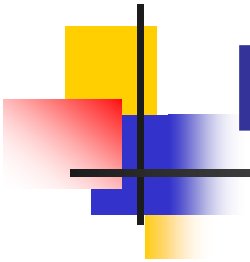


Постановка задачи тестирования на основе конечных автоматов

- Дана спецификационная модель и реализация, также построенная на основе конечного автомата (КА).
- Демонстрируют ли они эквивалентное поведение?
  То есть выдают ли на одну и ту же последовательность стимулов одну и ту же последовательность реакций?







Вопросы

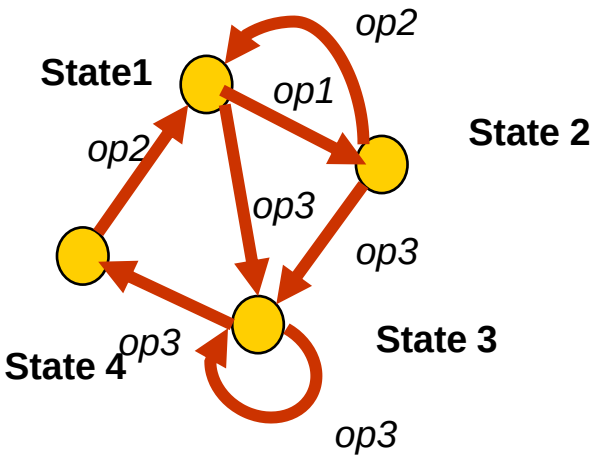
- Что является критерием эквивалентности (соответствия) двух конечных автоматов?
- Какие предположения важны при сравнении двух автоматов?
- Сколько тестов нужно для того, чтобы убедиться, что КА-модель и КА-реализация эквивалентны?
- Какую метрику тестового покрытия можно предложить?

Некоторые ответы

- Что является критерием эквивалентности (соответствия) двух конечных автоматов?
- Какие предположения важны при сравнении двух автоматов?
- Сколько тестов нужно для того, чтобы убедиться, что КА-модель и КА-реализация эквивалентны?
- Какую метрику тестового покрытия можно предложить?
  - Покрыть все состояния в модели/реализации
  - Покрыть все переходы …
  - Прокрыть все пути длиной 2 перехода …
  - **…**
- Задача об эквивалентности конечных автоматов: как построить различающие последовательности стимулов?

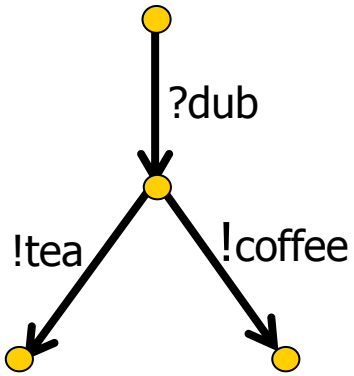
Некоторые ответы

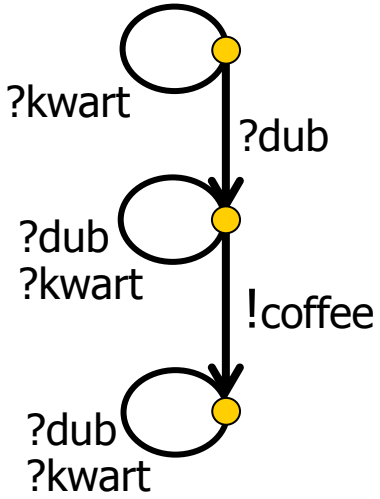
- Что является критерием эквивалентности (соответствия) двух конечных автоматов?
- Какие предположения важны при сравнении двух автоматов?
- Сколько тестов нужно для того, чтобы убедиться, что КА-модель и КА-реализация эквивалентны?
- Какую метрику тестового покрытия можно предложить?
  - Покрыть все состояния в модели/реализации
  - Покрыть все переходы …
  - Прокрыть все пути длиной 2 перехода …
  - **…**
- Задача об эквивалентности конечных автоматов: как построить различающие последовательности стимулов?



См. http://panda.ispras.ru/~kuliamin/

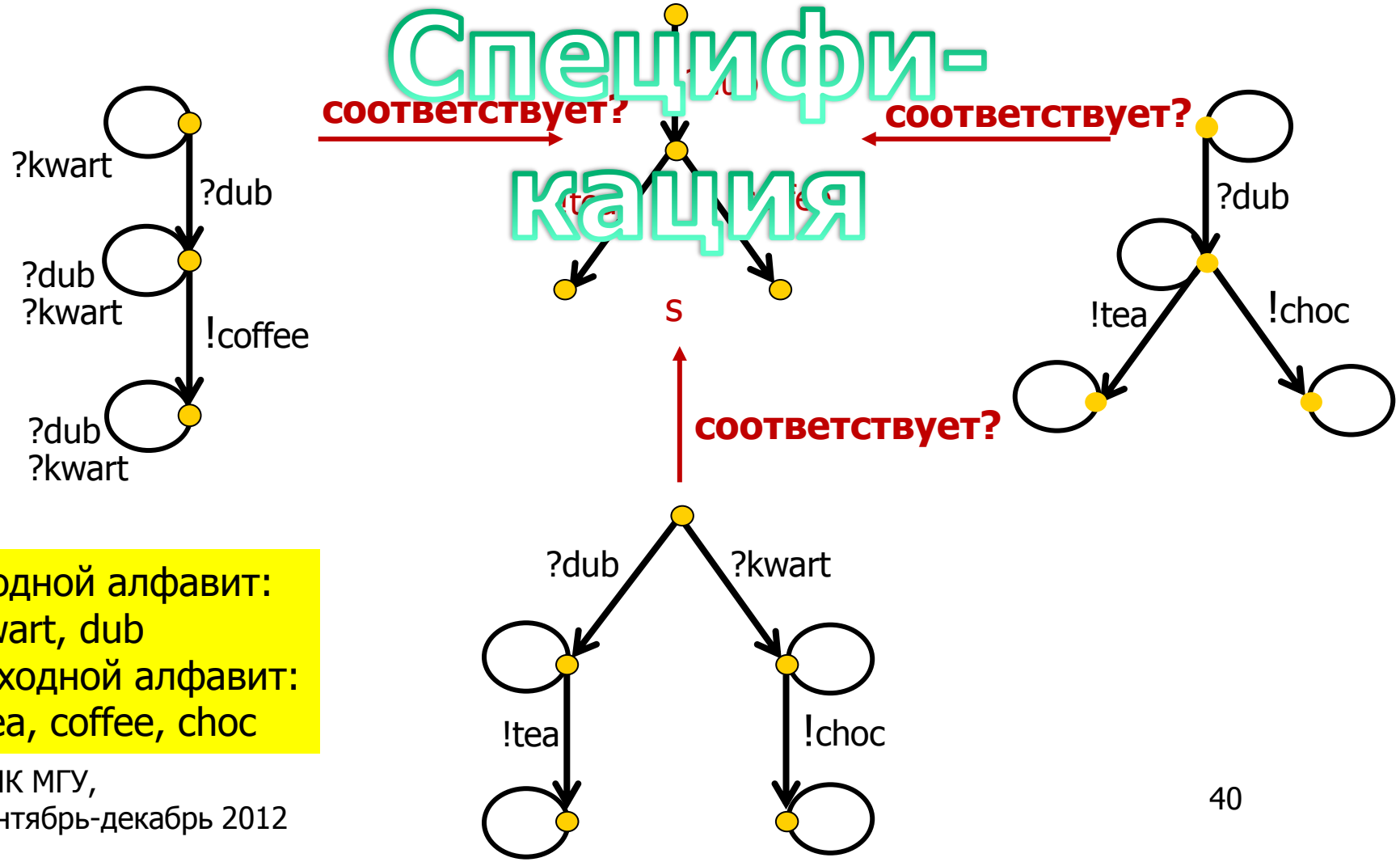
LTS – Labeled Transition System





- Возможны переходы из состояния в состояние при поступлении стимулов и выработке реакций.
- Возможен спорадический переход или переход по тайм-ауту.
- Возможна «неподвижность» системы – quiescence.

Анализ соответствия, когда спецификация задается в форме LTS, и возможны недетерминизм и неполная спецификация



Входной алфавит:
-kwart, dub
Выходной алфавит:
- tea, coffee, choc

Тестирование API программного модуля/объекта

- Если за основу формальной модели брать FSM или LTS, их размер и сложность могут оказаться существенно больше, чем размер и сложность реализации модуля.
- Спецификация программного контракта, как правило, компактнее реализации
- Вопрос: как построить тест на основе программного контракта?