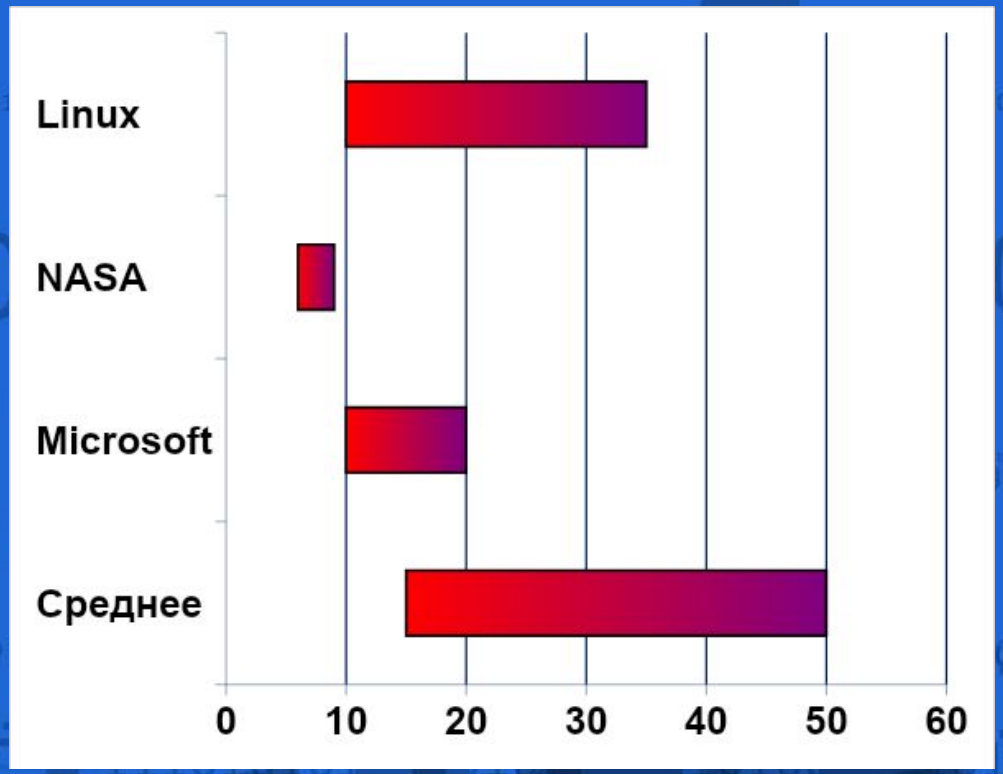
Корректность программ

В. В. Кулямин

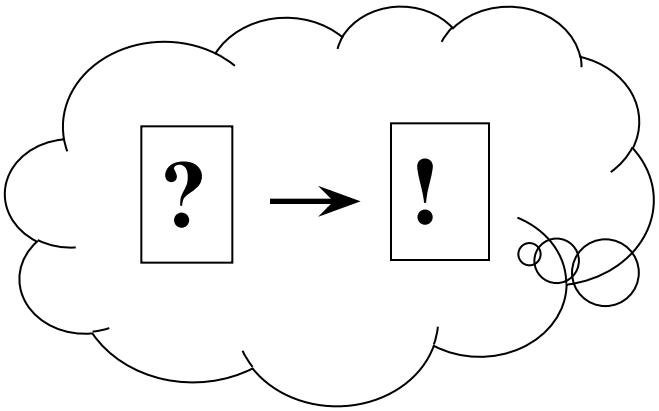
Институт системного программирования РАН

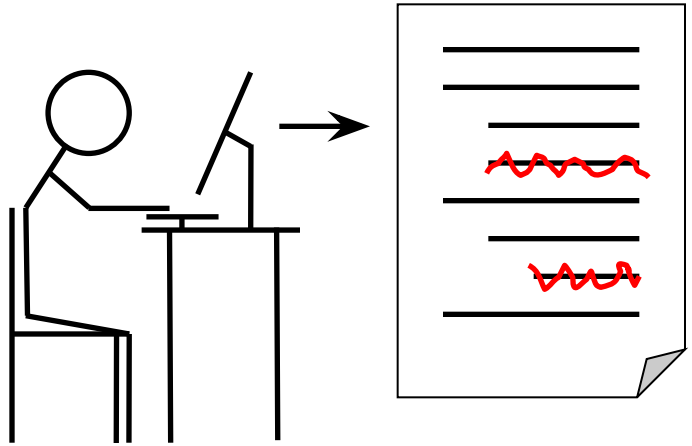
Статистика ошибок

Среднее количество ошибок
на 1000 строк кода до тестирования



Причины ошибок





- Неправильное понимание задач
- Неправильное решение задач
- Неправильный перенос решений в код

Сложность программ

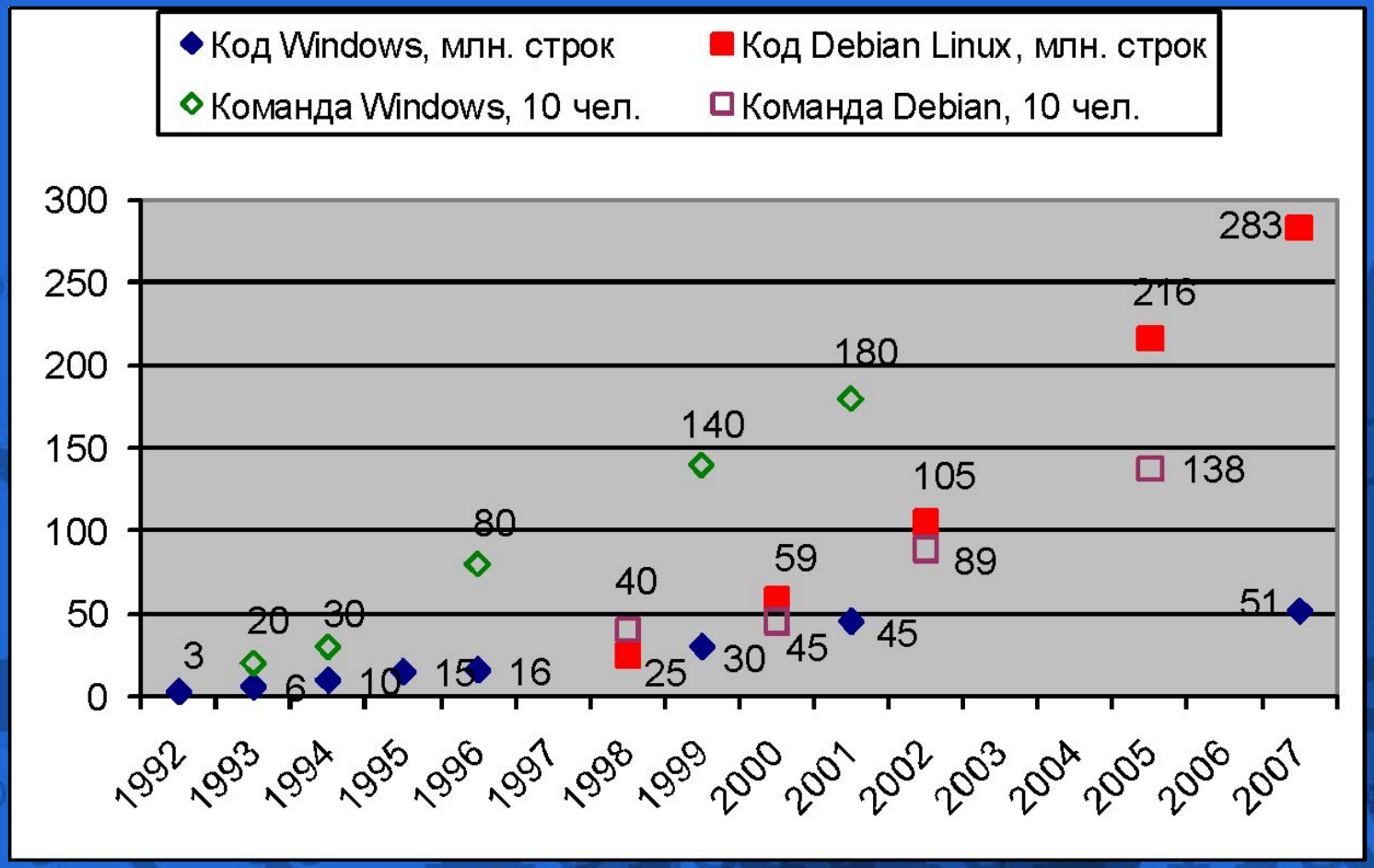
Основная причина ошибок в ПО –

его **сложность**

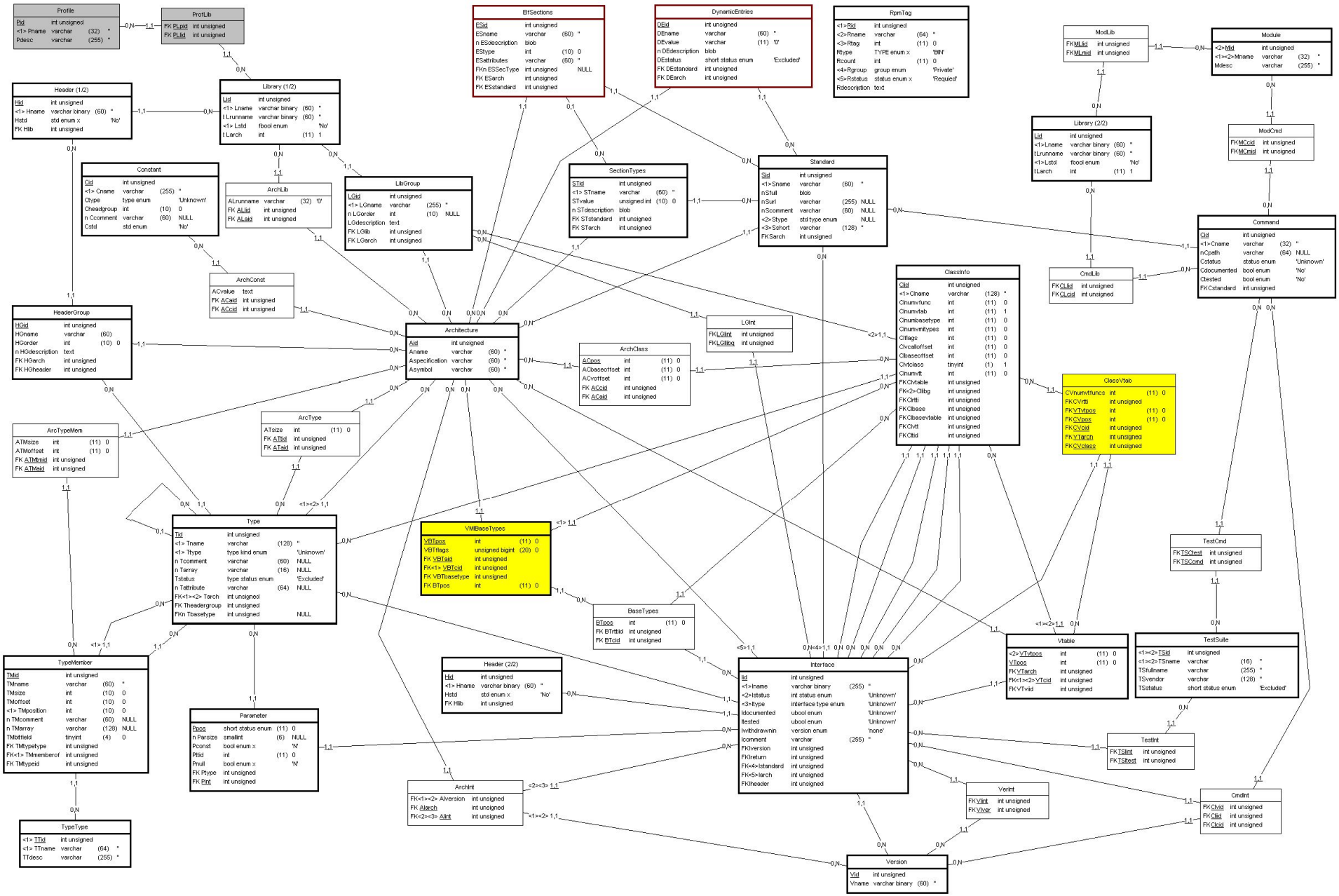
Конференция NATO 1968 –

software сложнее hardware

Сложность – большой размер



Сложность сложность данных



Сложность задач и интерфейса

Сложность – запутанность

```
int unknown_f(int x0, int x1) {
  if(x0 == 0)  return x1;
  if(x1 == 0)  return x0;
  if(x0>0 && x1<0 || x0<0 && x1>0) x1 = -x1;
  while(x1 != 0) {
    if(x1>x0 && x0>0 || x1<x0 && x0<0)
    { x0 = x1-x0; x1 = x1-x0; x0 = x0+x1; }
    x1 = x0-x1;
    x0 = x0-x1;
  }
  return x0;
}
```

Борьба с ошибками

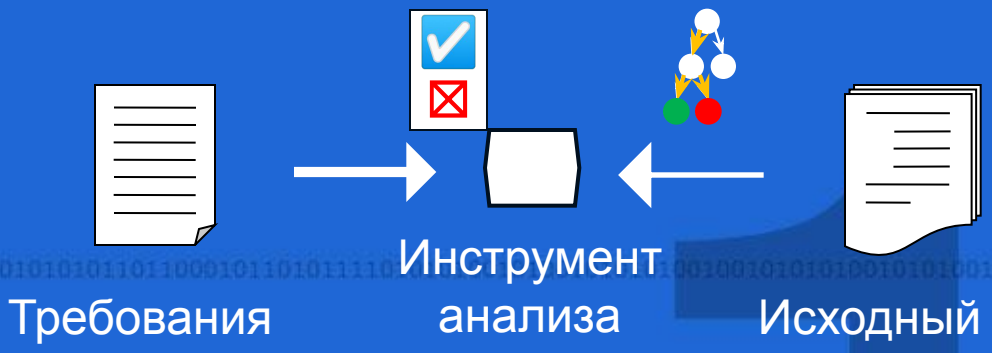
- Безошибочное программирование
  - Сильно ограничивается сложностью
- Автоматизация разработки
  - Повышает сложность возможных систем
  - Изменяет существенные источники ошибок
- Интеграция разработки и контроля качества
  - Нужны методы и инструменты контроля качества ПО
- Стандартизация
  - Нужны методы и инструменты проверки соответствия стандартам

Контроль качества ПО

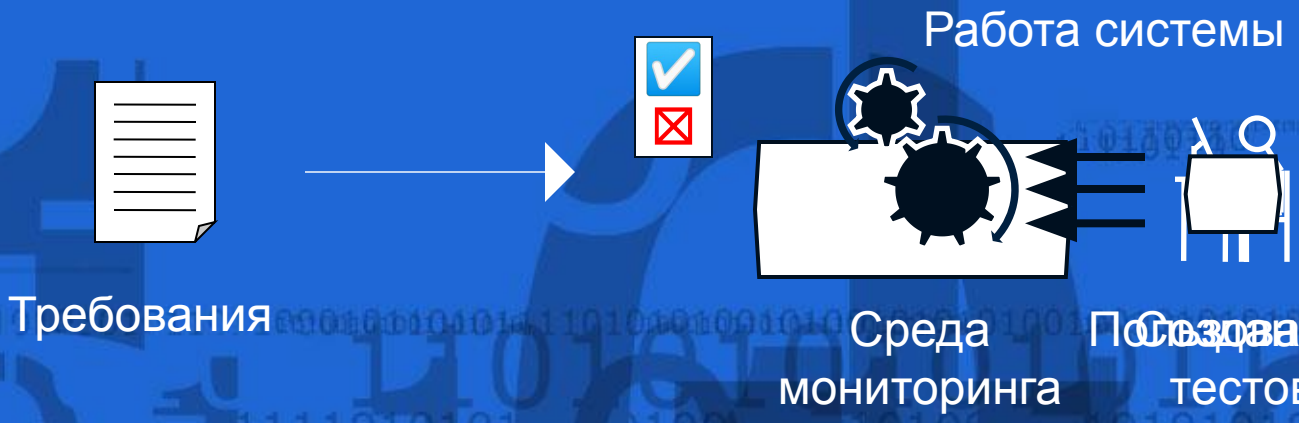
- Экспертиза (review, inspection)
- Статический анализ
  - Проверка правил корректности
  - Поиск конкретных ошибок по шаблонам
- Динамический анализ
  - Мониторинг
  - Тестирование
- Формальная верификация
  - Дедуктивный анализ
  - Проверка моделей
- Гибридные методы

Статика и динамика

- Статический анализ



- Динамический анализ

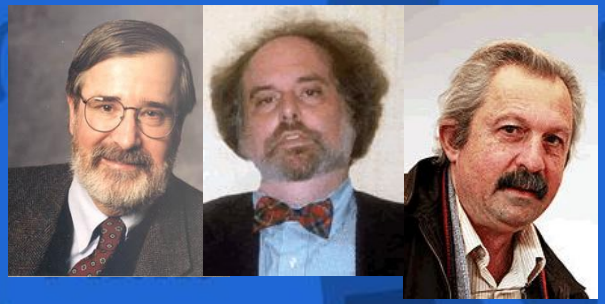


Формальная верификация

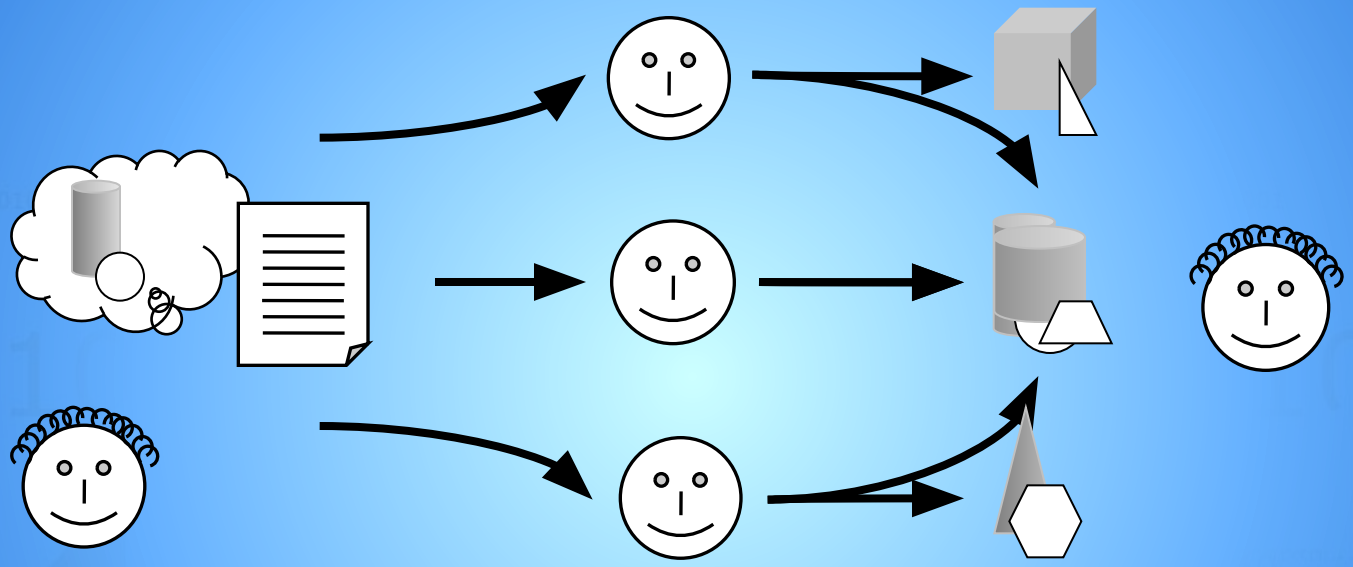
- Дедуктивный анализ
  [R. Floyd 1967, C. A. R. Hoare 1969]
  - Логика Хоара – {Pre} Program {Post}
  - Правила вывода

$$\frac{\{B \wedge P\}\ S\ \{Q\}\ ,\ \{\neg B \wedge P\}\ T\ \{Q\}}{\{P\}\ \textbf{if}\ B\ \textbf{then}\ S\ \textbf{else}\ T\ \textbf{endif}\ \{Q\}}$$

- Проверка моделей
  [E. M. Clarke & E. A. Emerson 1980,
  J. P. Queille & J. Sifakis 1982]
  - Анализ достижимых состояний



Зачем нужна формальность?



Гибридные методы

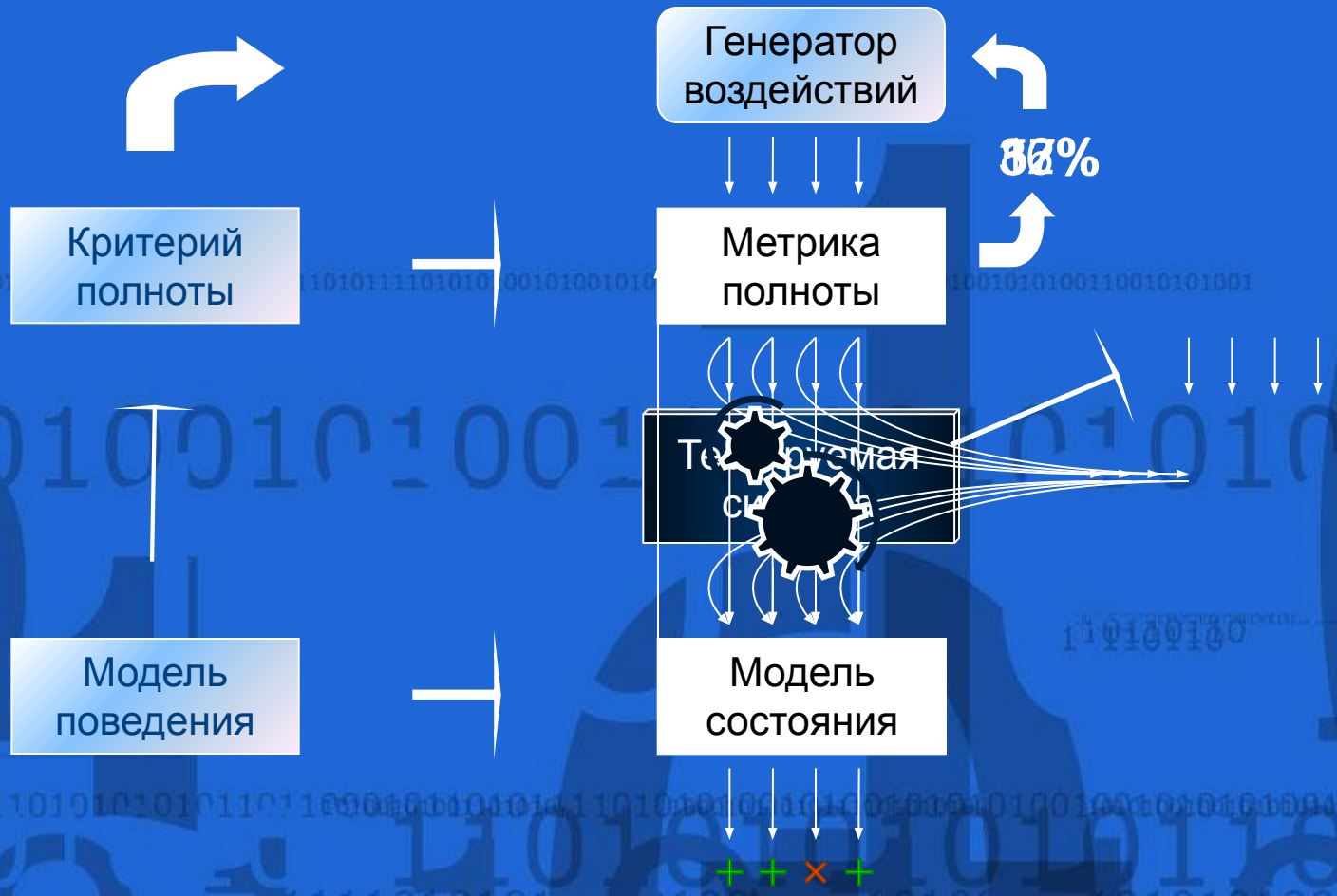
Интегрируют элементы различных подходов

- Тестирование на основе моделей
- Расширенный статический анализ
- Формальный мониторинг
- Синтетическое структурное тестирование

Вспомогательные техники

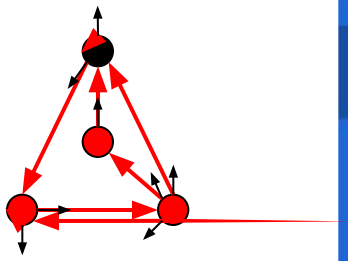
- Символическое исполнение
- Абстрактная интерпретация
- Вывод ограничений
- Разрешение ограничений
- Автоматическое уточнение моделей

Тестирование на основе моделей



Пример: описание и работа теста

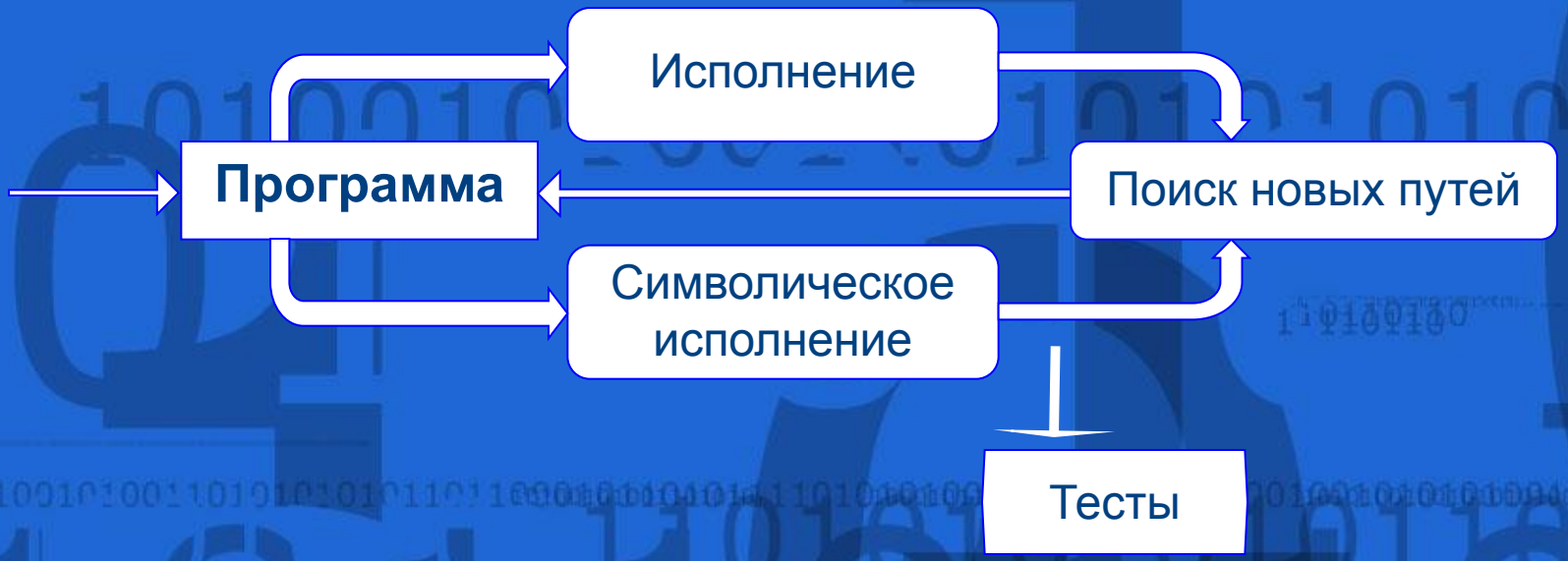
```
@Test public class AccountTest
{
  Account account;

  @State public int getBalance()   { return account.getBalance(); }

  @Test @DataProvider(name = "sums") @Guard(name = "bound")
  public void testDeposit(int x)   { account.deposit(x); }

  @Test @DataProvider(name = "sums")
  public void testWithdraw(int x)  { account.withdraw(x); }

  public boolean bound()           { return getBalance() < 500; }

  public int[] sums = new int[]{0, 1, 137, 1000000};
}
```



Синтетическое тестирование

DART

[P. Godefroid, G. Agha, K. Sen 2005]



Пример работы DART

```
int unknown_f(int x0, int x1) {
  if(x0 == 0)   return x1;
  if(x1 == 0)   return x0;
  if(x0>0 && x1<0
     || x0<0 && x1>0)  x1 = -x1;
  while(x1 != 0) {
    if(x1>x0 && x0>0
       || x1<x0 && x0<0)
    { x0 = x1-x0; x1 = x1-x0;
      x0 = x0+x1; }
    x1 = x0-x1;
    x0 = x0-x1;
  }
  return x0;
}
```

x1' = 0

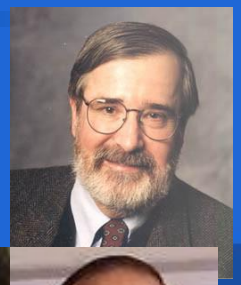
x0' = 1

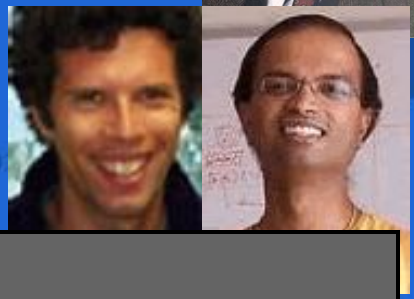
x0	x1	Ограничение
0	0	x0 == 0
1	0	!(x0 = 0) && x1 == 0
1	1	!(x0 = 0) && !(x1 == 0) && !(x0>0 && x1<0 || x0<0 && x1>0) && !(x1>x0 && x0>0 || x1<x0 && x0<0)
1	5	!(x0 = 0) && !(x1 == 0) && !(x0>0 && x1<0 || x0<0 && x1>0) && (x1>x0 && x0 >0 || x1<x0 && x0 < 0)

Авто-уточнение моделей

Counterexample guided abstraction refinement

[E. M. Clarke & O. Grumberg et al 2000,
T. Ball & S. K. Rajamani 2000]





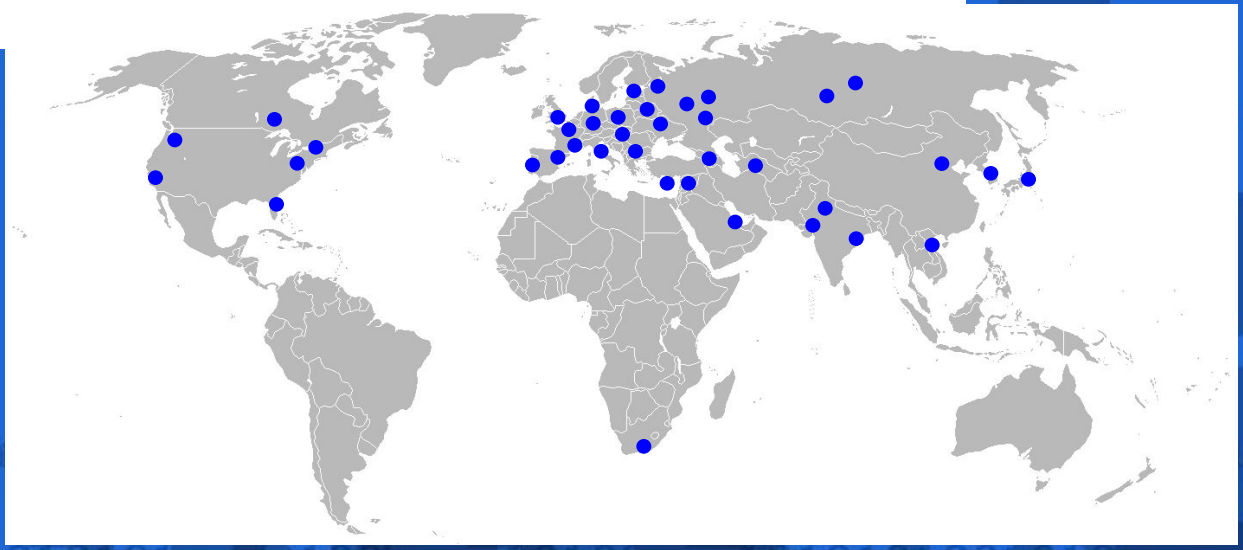
```
do {
  nPacketsOld = nPackets;
  ...
  if(request) {
  ...
    nPackets++;
  }
} while (nPackets != nPacketsOld);
```

```
do {
  b = true;
  ...
  if(*) {
  ...
    b = b?false:*;
  }
} while (!b);
```

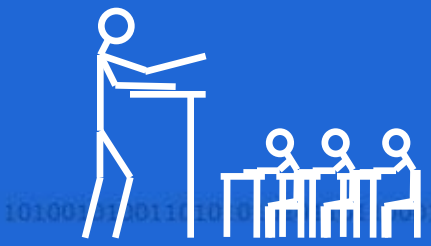
Работы отдела ТП

- **Разработка тестов и тестирование**
  - Информационная система оператора связи
  - Операционные системы реального времени
  - Базовые библиотеки Linux (Linux Standard Base)
  - Протоколы IPv6, Mobile IPv6, IPsec
  - Отдельные модули компиляторов Intel
  - Микропроцессоры архитектуры MIPS
- **Создание технологий и инструментов**
  - Тестирование на основе моделей (UniTESK)
  - Проверка соответствия стандарту LSB
  - Верификация драйверов Linux

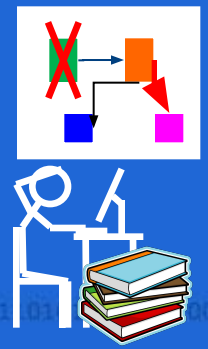
Разработки и исследования



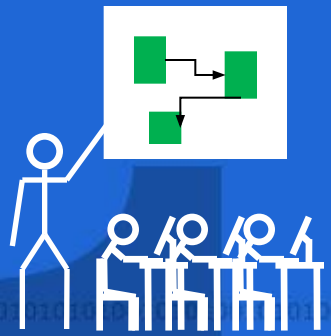
Карьера в ИСП РАН



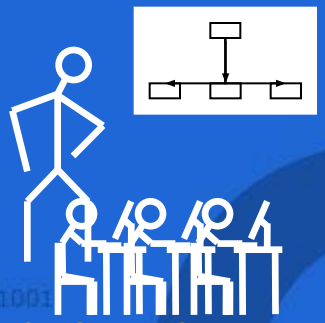
преподаватель



исследователь



архитектор



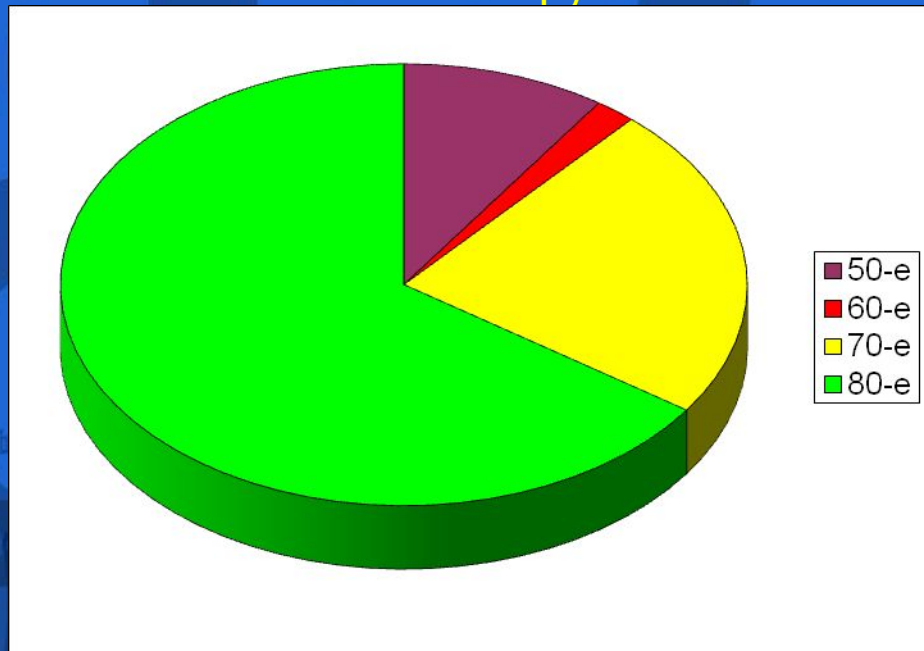
руководитель группы

старший разработчик

разработчик

аспирант

студент



Спасибо за внимание!

Вопросы?