# Интегрированная методика автоматизированного построения формальных поведенческих моделей С-приложений по исходному коду

Юсупов Юрий Вадимович

Специальность 05.13.11 –
Математическое и программное обеспечение
вычислительных машин, комплексов и компьютерных сетей

Научный руководитель: проф. кафедры ИУС, ФТК

Котляров Всеволод Павлович

# Особенности промышленной разработки программного обеспечения

- постоянный рост требований к качеству производимого ПО
- борьба за качество начинается на самых ранних этапах разработки ПО и заключается в нахождении и исправлении ошибок в первых версиях программных продуктов
- обеспечение необходимого уровня качества только за счет динамической проверки (тестированием) правильности функционирования ПО становится невозможным
- переиспользование старого кода
- восстановление документации и ее поддержка в актуальном виде

# Цели и задачи исследования

**Цель** – разработка методики автоматизированного построения формальных поведенческих моделей С-приложений по исходному коду, пригодных для статического и визуального анализа поведенческих и структурных свойств.

**Задачи:**

- анализ области автоматизированного построения формальных моделей по их исходному коду на основе сравнительного анализа промышленных инструментов возвратного проектирования и формальных нотаций;
- определение модели поведения для систем, реализованных на языке С, пригодной для статического и визульного анализа;
- создание методик формализации фрагментов исходного кода С-приложений с помощью выбранной формальной нотации;
- разработка программной реализации, позволяющей обеспечить генерацию формальных спецификаций по фрагментам исходного кода С-приложений;
- внедрение разработанной методики и программных средств в процесс производства и поддержки ПО.

# Область исследования

Возвратное проектирование –

> “это процесс анализа системы с целью идентификации системных компонентов и их взаимодействий (поведенческих свойств) и создания представления системы в другой форме или на более высоком уровне абстракции”. (E. Chikofsky, J. Cross)

- Цели возвратного проектирования:
  - создание альтернативных форм описания системы для облегчения понимания и повышения уровня осмысления;
  - восстановление утраченной информации о системе с целью восстановления документации;
  - построение моделей программ с целью верификации и тестирования.
- Методы возвратного проектирования:
  - Статический анализ.
  - Динамический анализ.

# Инструментарий возвратного проектирования

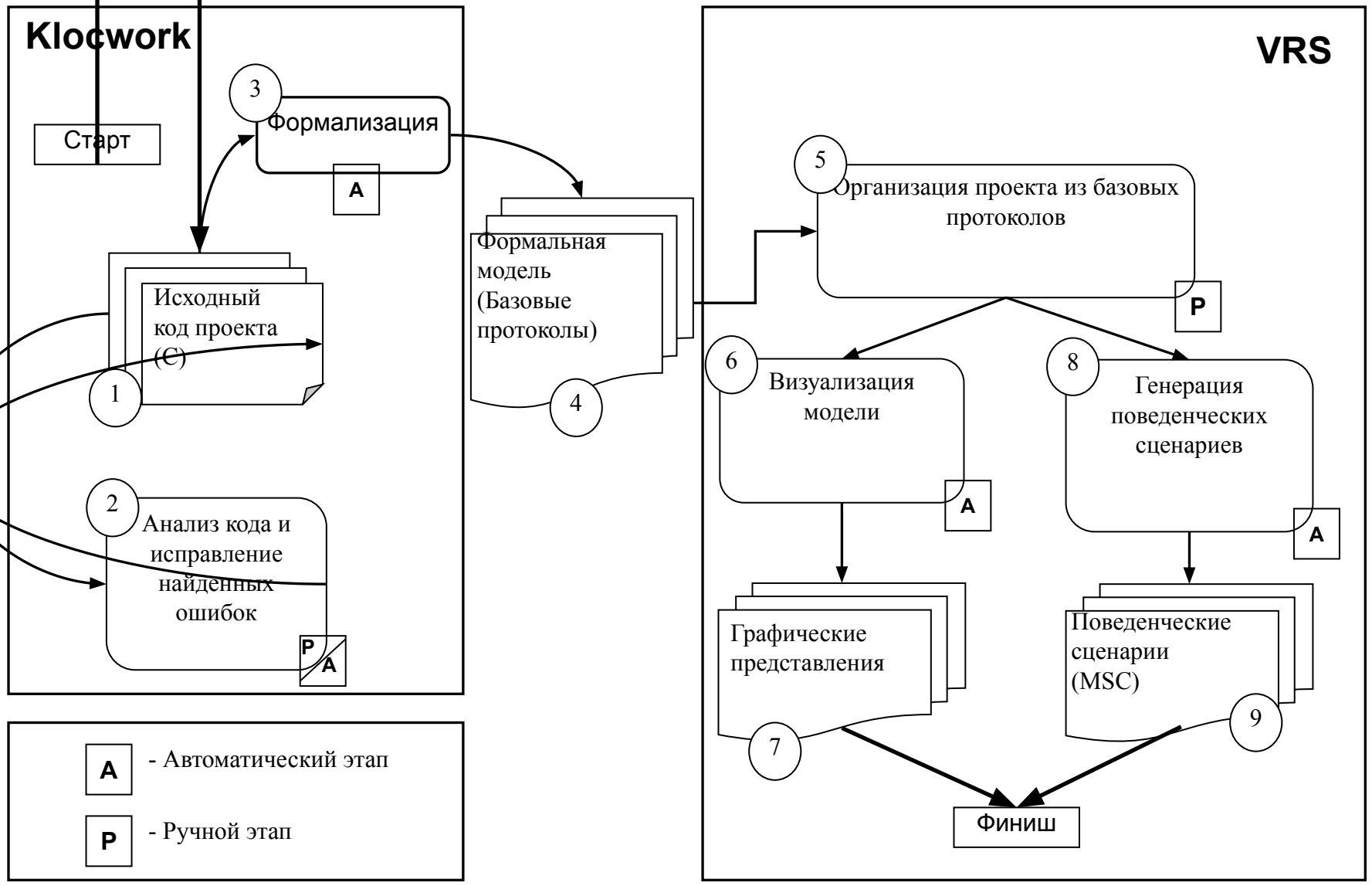
| | Klocwork Insight | Source-Navigator | CodeSurfer | CC-Rider | Imagix 4D | Cristal FLOW |
|---|---|---|---|---|---|---|
| Целевая платформа | Win32, Unix | Win32, Unix | Win32, Unix | Win32, Unix | Win32, Unix | Win32 |
| Поддержка языков | C, C++, C#, Java | C, C++, Java, Tcl, FORTRAN,COBOL | C, C++ | C, C++ | C, C++, Java | C, C++ |
| **Объем поддерж. кода (MLOC)** | **>5** | **0.5** | **0.5** | **1** | **<1** | **1** |
| Генерация документации | HTML, PDF | TEXT | Нет | RTF, HTML, WinHelp, GWSC | RTF, HTML,TEXT | HTML |
| мостр. формальн. моделей* | Нет | Нет | Нет | Нет | Нет | Нет |
| Визуализация графов зависимостей | CallGrap, CFG | CallGraph | CallGraph, CFG, CDG, DDG, CDG, SDG | CallGraph | CallGraph, CFG, DDG | CallGraph, DDG, CFG |
| **Расширение баз. функцион. (язык прогр-я)** | **Да (C,C+)** | **Да (Tcl)** | **Да (SCHEME)** | **Да (C,C++)** | **Нет** | **Нет** |
| **Доступ к внутр. представл. кода** | **Да** | **Да** | **Да** | **Да** | **Нет** | **Нет** |
| Сбор метрик по коду | Да | Нет | Да | Да | Да | Да |

* Пригодных для статического и визуального анализа автоматизированными средствами

# Сравнительный анализ формальных нотаций

| | Критерии | | | | | | |
|---|---|---|---|---|---|---|---|
| | **Теоретич. модель** | **Инструмент. поддержки** | **Текстовое предст-е** | **Графич. предст-е** | **Композ./ декомпоз.** | **Мин. един. описания** | **Поведенческ. сценарии** |
| **MSC** | CSP | Telelogic SDL TTCN, Telelogic TAU G2, VRS | + | - | - | Сценарий | - |
| **SDL** | CFSM | Telelogic SDL TTCN | + | + | + | Кон.-авт. диаграмма | + |
| **UML** | CFSM | Telelogic TAU G2 | + | + | + | Кон.-авт. диаграмма | + |
| **Basic protocols** | ATS | VRS | + | + | + | Базовый протокол | + |
| **Promela** | CSP | Spin | + | - | - | Процесс | + |
| **Lotos** | CCS, CSP | LOTOS | + | - | - | Процесс | - |
| **Estelle** | CNSM | EDT | + | - | - | Кон.-авт. диаграмма | + |
| **VDM-SL** | LPF | VDM | + | - | - | Функция | - |
| **RSL** | VDM-SL, CSP | RAISE | + | - | - | Процесс | - |

# Концепция предлагаемого подхода



# Аппарат описания модели поведения программной системы

***Атрибутная транзиционная система***

***< S, A, T, L, l >***

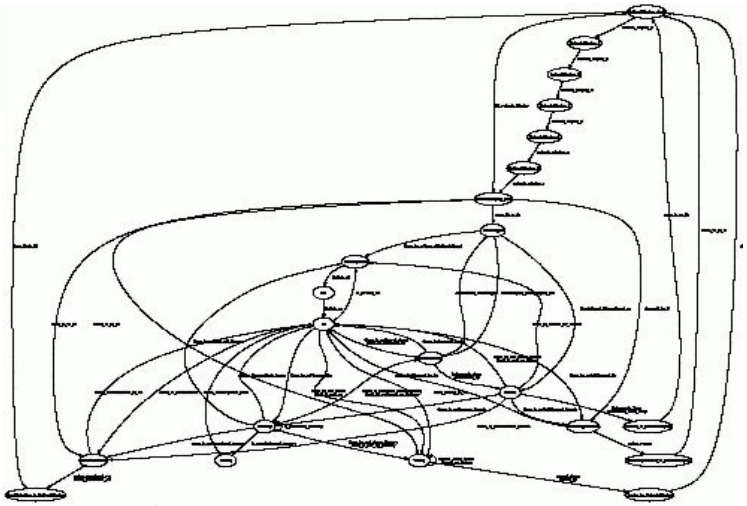
– ***S*** – множество состояний;

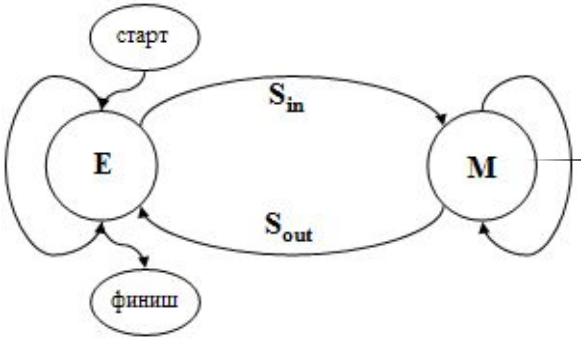
– ***A*** – множество действий;

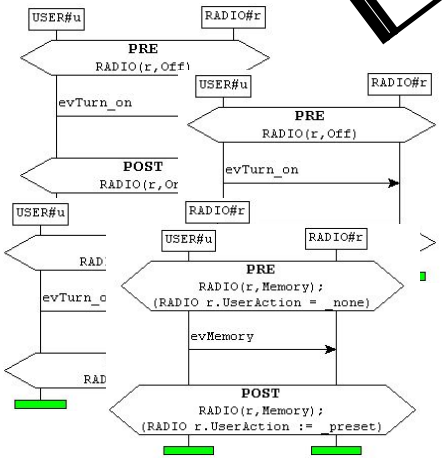
– ***T*** – множество размеченных переходов $(s \xrightarrow{a} s')$ и неразмеченных (скрытых) переходов $(s \to s')$

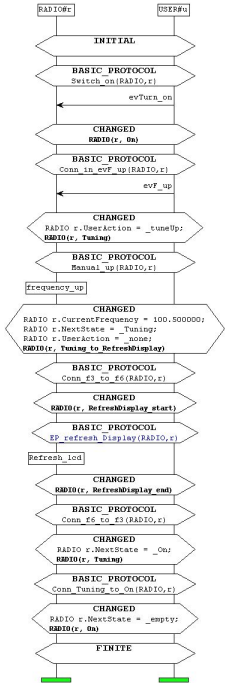
– ***L*** – множество атрибутных разметок;

– $l : S \to L$ – частично определенная функция разметки состояний.






E = ($e_1$, $e_2$, $e_3$,…)
M = ($m_1$, $m_2$, $m_3$,…)

$S_{in}$ = ($in_1$, $in_2$, $in_3$,…)
$S_{out}$ = ($out_1$, $out_2$, $out_3$,…)

# Динамические аспекты модели поведения

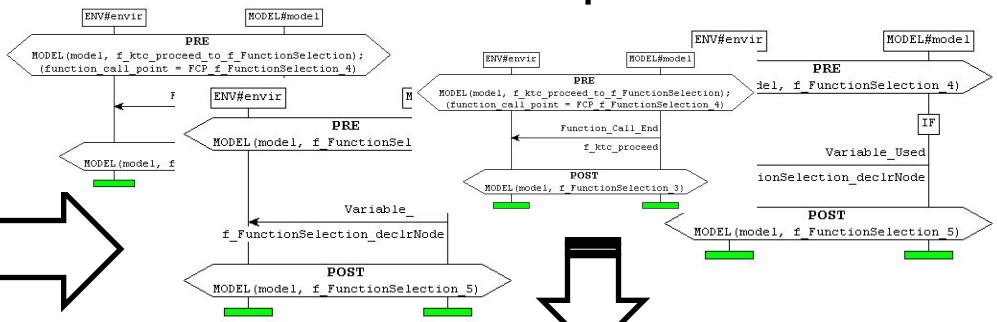
## 1. Исходный код

...

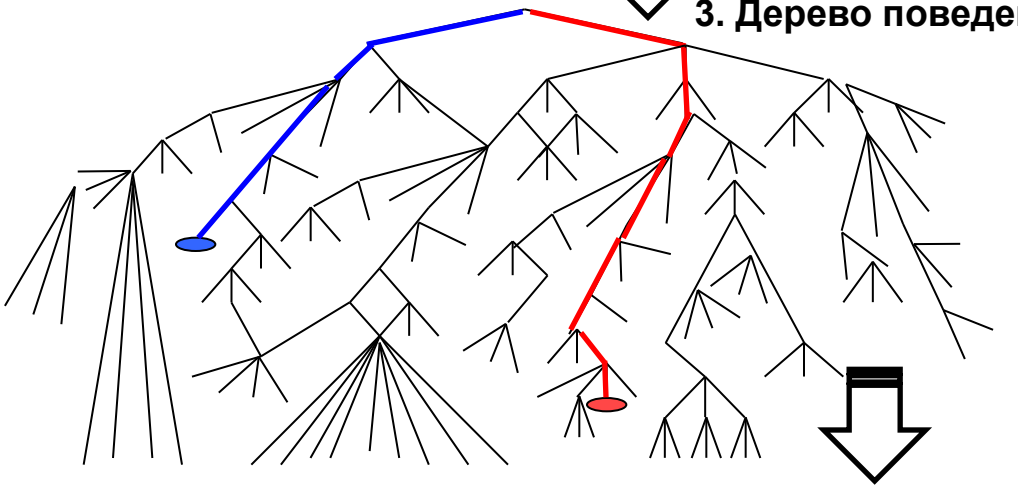
```
 int  pid;
 int  replyEvent;
 char *replyMessage;
 void *replyPointer;
 int  len;
 int  notification = 0; /* not notification by default */
 char * rpcmode = ""; /*no extra text by default */

 if (argc > 1 && strcmp (argv[1], "-listpm") == 0) {
   /* Get all postmaster! */
   int bufferPid[100];
   char* bufferText[100];
   int noOfPM;
   int i;
   noOfPM = SPFindActivePostMasters (bufferPid,
bufferText, 100);
   for (i = 0; i < noOfPM; i++) {
     if (bufferText[i] != NULL) {
       printf ("Pid: %d, Created: %s\n", bufferPid[i],
bufferText[i]);
       SPFree (bufferText[i]);
     }
   }
   /* break when ready*/
   exit (0);
 }
 if ( argc > 3 && !strcmp( argv[argc-1], "-notification" ) ) {
   notification = 1;
   rpcmode = "NOTIFICATION";
   argc--;
   argv[argc] = 0; /* hide this flag to avoid later confusion
*/
 }

 if (argc < 3) {
   printf ("usage: %s <tool> <event> [<data>...]
[-notification]\n", argv[0]);
   exit (1);
```

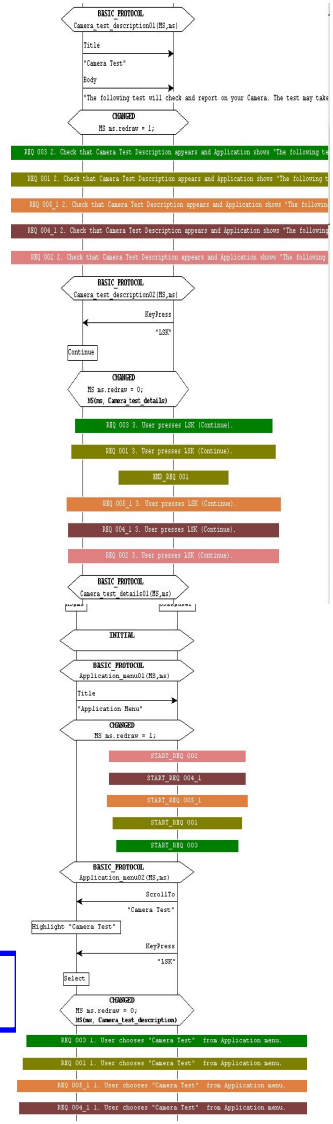
...

## 2. Базовые протоколы




## 3. Дерево поведения





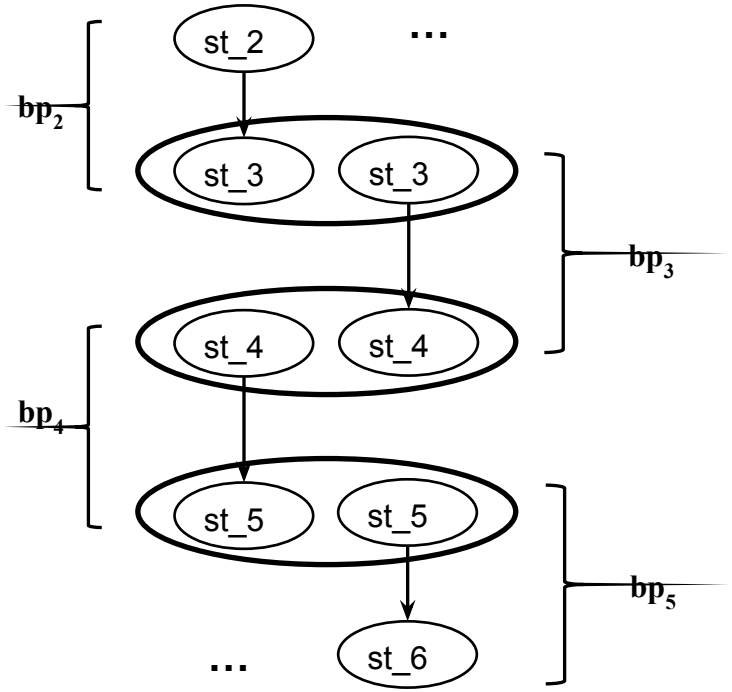
## 4. Поведенческие сценарии

$$s1 \xrightarrow{bp_1} s2 \xrightarrow{bp_3} s5 \xrightarrow{bp_7} s9 \xrightarrow{bp_{14}} s35 \xrightarrow{bp_{56}} s58$$

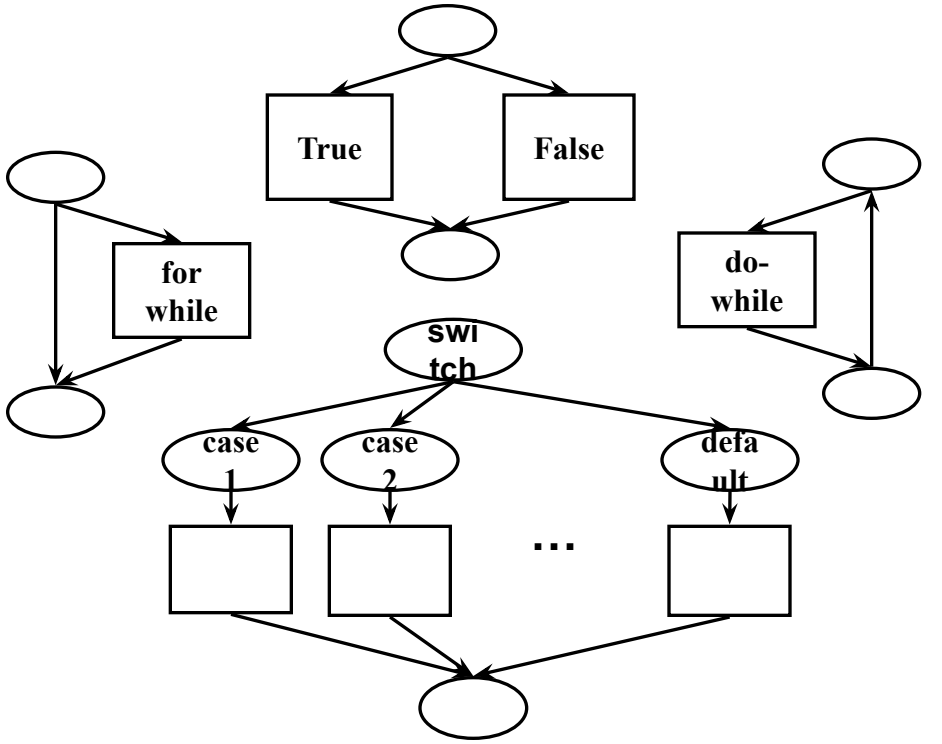
$$s1 \xrightarrow{bp_2} s3 \xrightarrow{bp_5} s4 \xrightarrow{bp_8} s13 \xrightarrow{bp_{21}} s22 \xrightarrow{bp_{34}} s49 \xrightarrow{bp_{59}} s62 \xrightarrow{bp_{61}} s75 \xrightarrow{bp_{64}} s81$$

# Методика 1: сохранение потока управления программы

**Связь базовых протоколов по состояниям агента-приложения**



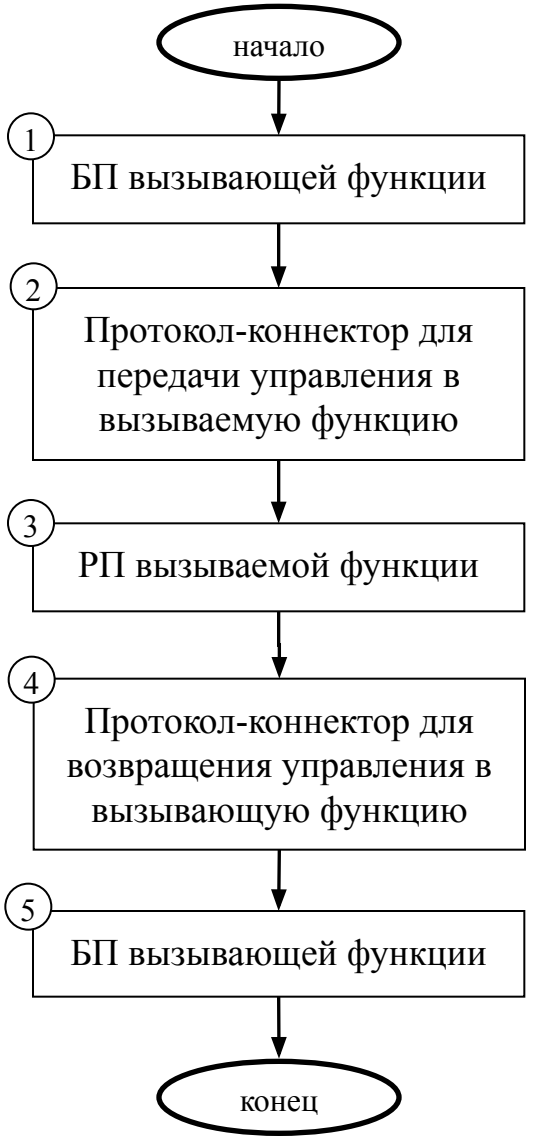

**Фрагменты систем переходов для нелинейных фрагментов кода**




$$...st_2 \xrightarrow{bp_2} st_3 \xrightarrow{bp_3} st_4 \xrightarrow{bp_4} st_5 \xrightarrow{bp_5} st_6...$$

st_2, st_3, st_4, st_5, st_6 – состояния агента-приложения;
$bp_2$, $bp_3$, $bp_4$, $bp_5$ – базовые протоколы.

# Методика 2: формализация вызовов функции (1)




1) $bp_n=(a_{n-1}, a_n)$

$(a_{n-1}, a_n) \in A$

2) $cp_1=(a_n,b_1)$

$(a_n) \in A$

$(b_1) \in B$

3) $ep=(b_1, b_m)$

$(b_1, b_m) \in B$

4) $cp_2=(b_m, a_{n+1})$

$(b_m) \in B$

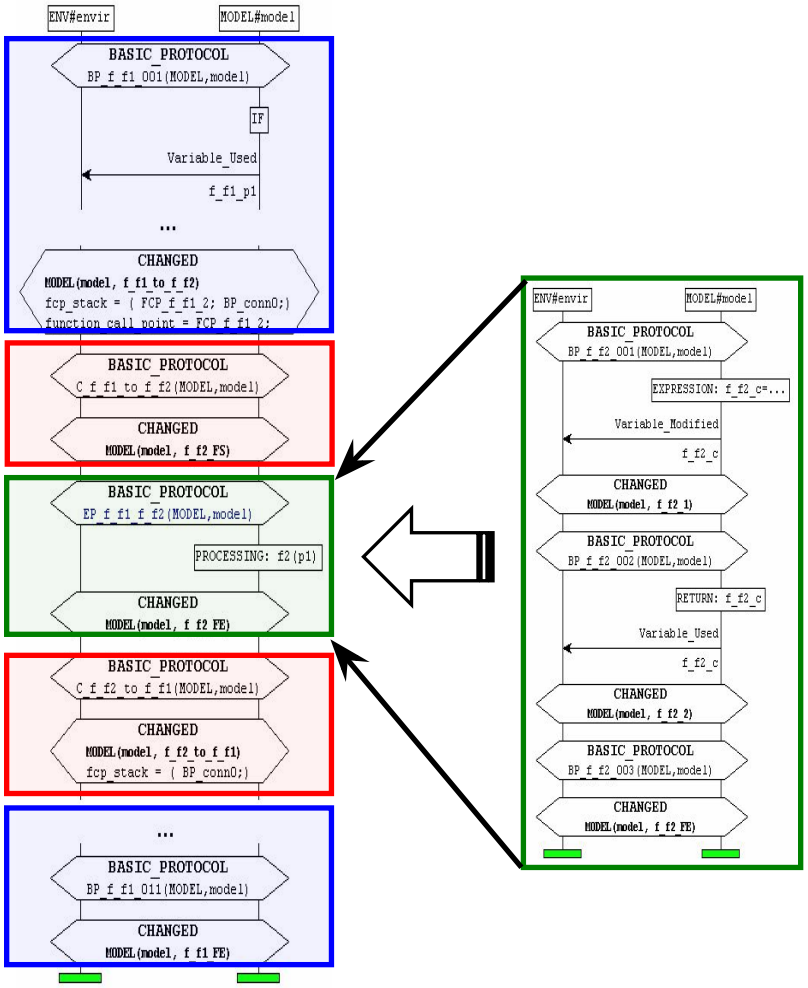
$(a_{n+1}) \in A$

5) $bp_{n+1}=(a_{n+1}, a_{n+2})$

$(a_{n+1}, a_{n+2}) \in A$

A – множество состояний вызывающей функции

B – множество состояний вызываемой функции

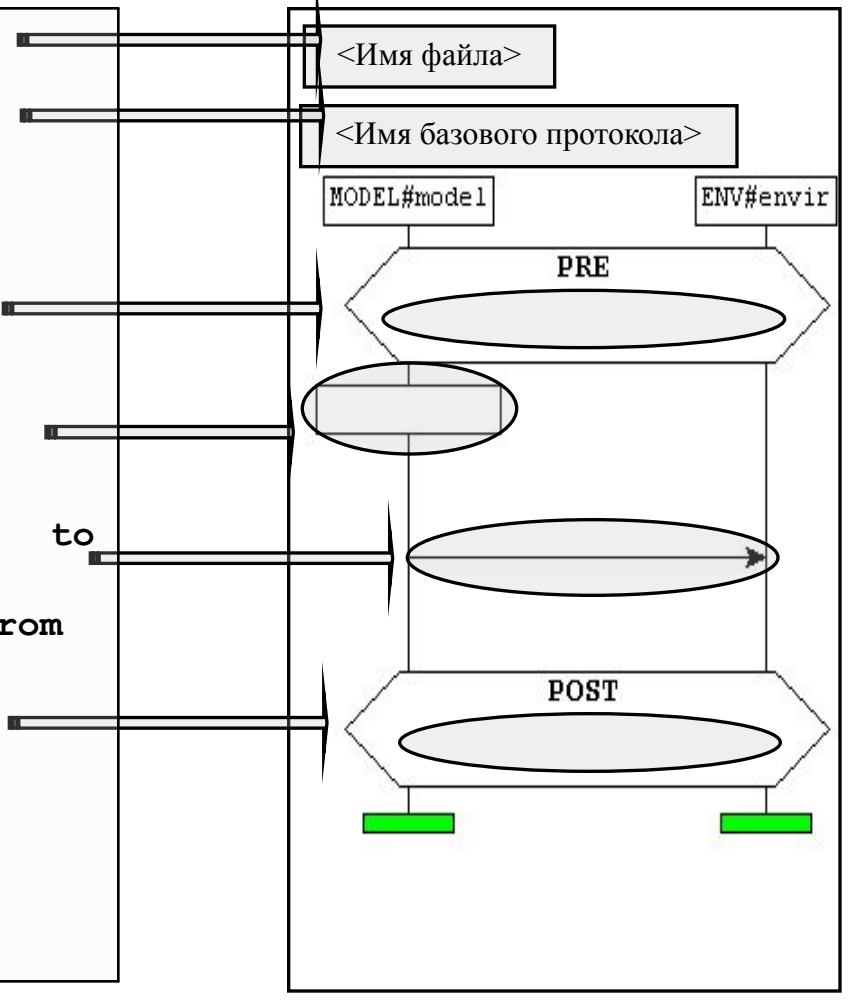


# Методика 3: построение базовых протоколов

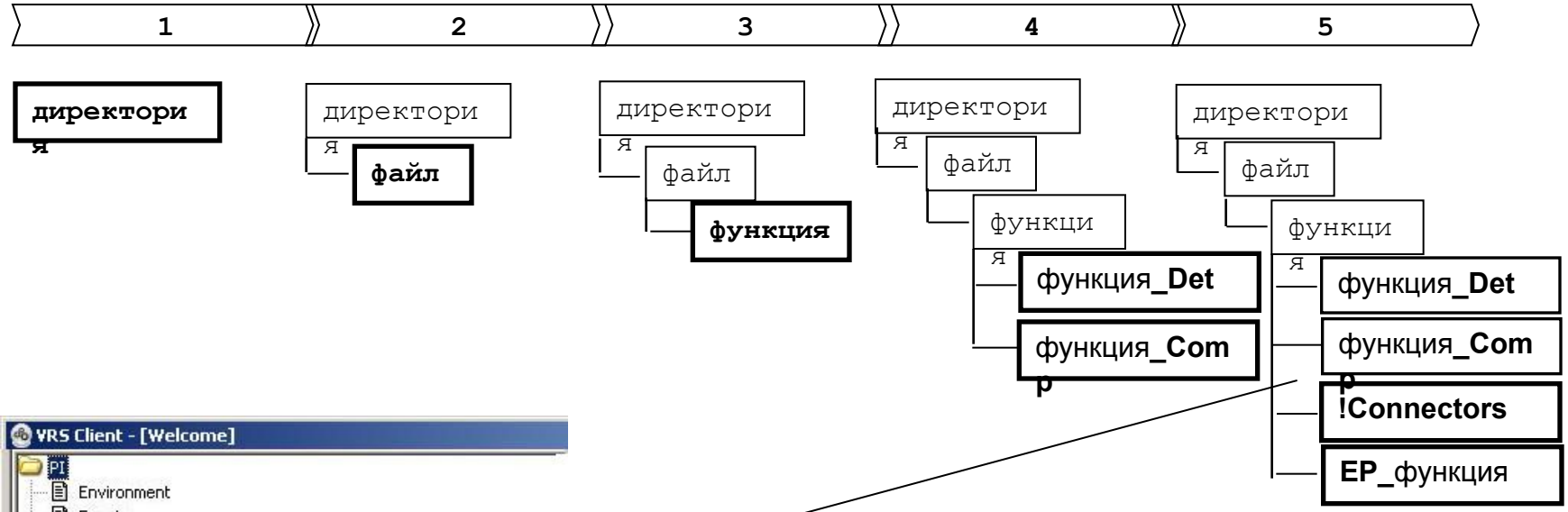
**Текстовое представление шаблона MSC диаграммы**

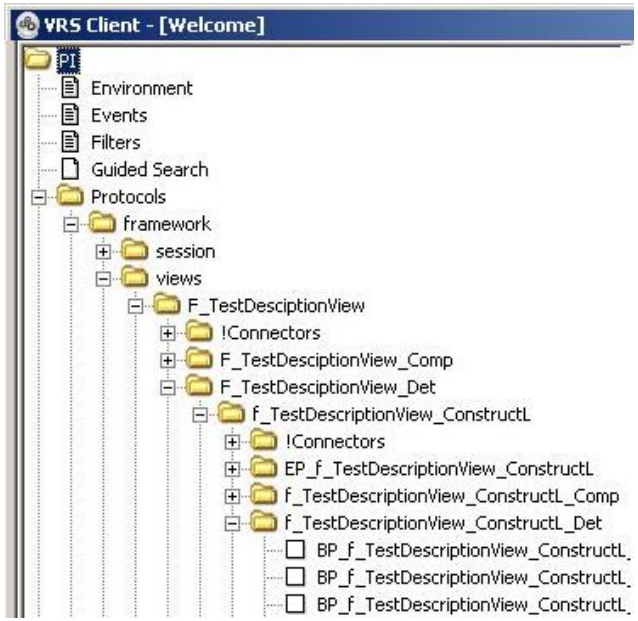
```
mscdocument <имя файла>;
msc <имя базового протокола>;
ENV#envir: instance;
MODEL#model: instance;

all: condition PRE /*MODEL(model,
<состояние агента>);
<атрибуты и параметры агента>*/;

MODEL#model: action '<действие>';

MODEL#model:   out   <сигнал>(<параметры>)   to
ENV#env;
ENV#env:     in     <сигнал>(<параметры>)from
MODEL#model;

all: condition POST /*MODEL(model,
<состояние агента>);
<атрибуты и параметры агента>*/;

ENV#envir: endinstance;
MODEL#model: endinstance;
endmsc;
```

**Графическое представление шаблона MSC диаграммы**



# Методика 4: структурирование базовых протоколов




– **функция_Det** – базовые протоколы, описывающие поведение функции на детальном уровне

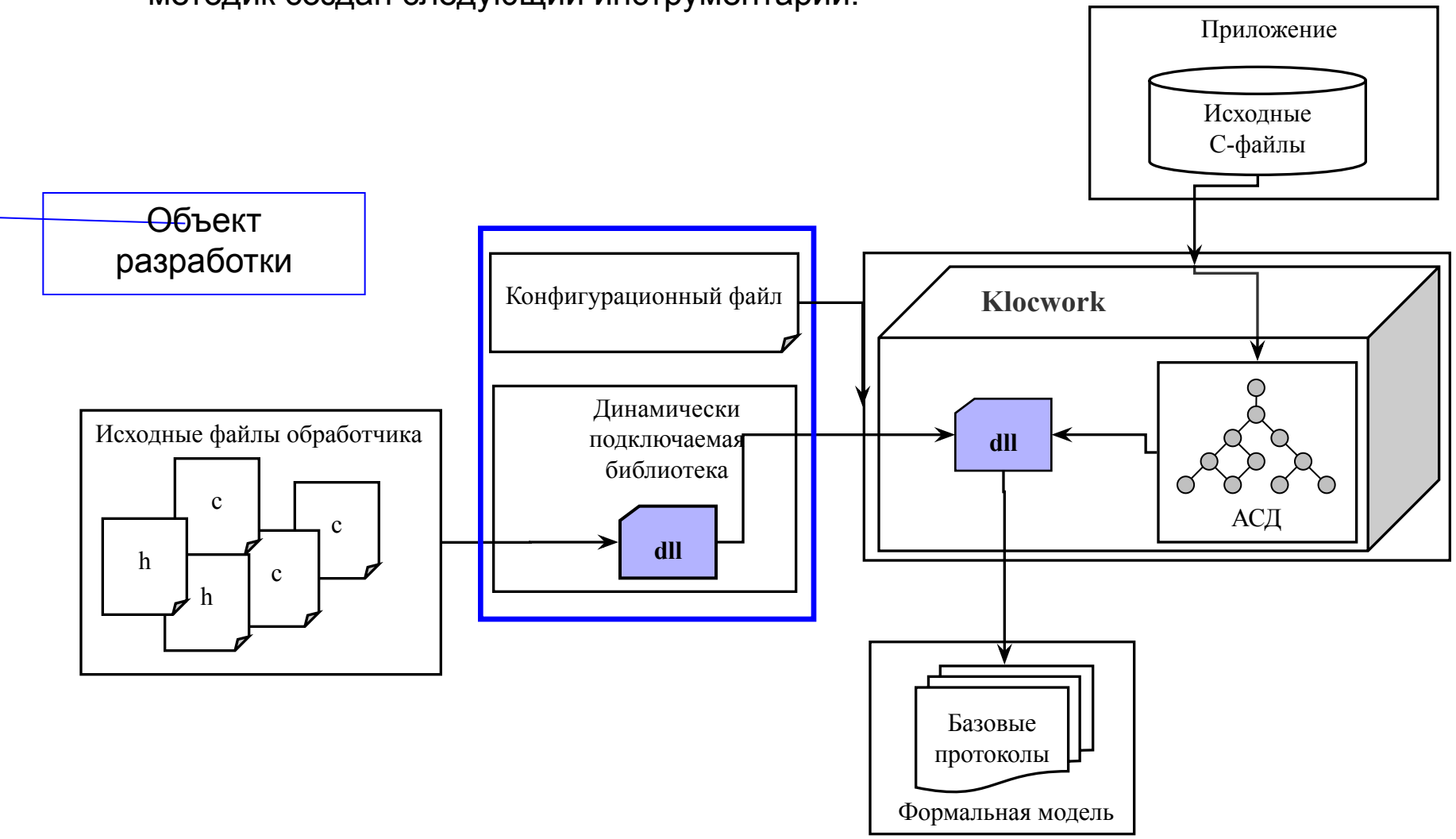
– **функция_Comp** – базовые протоколы, описывающие поведение функции на некотором уровне абстракции

– **!Connectors** – протоколы-коннекторы для моделирования вызовов функций

– **EP_ функция** – расширенные протоколы, описывающие поведение вызываемых функций

# Программная поддержка

В рамках работы для решения поставленных задач и реализации разработанных методик создан следующий инструментарий:



# Метрика оценки объема модели

$$M = \sum_{i=1}^{k} \left( BP^i + EP^i + CP^i \right)$$

- ***k*** – количество функций в проекте;
- ***BP*** – количество базовых протоколов, кодирующих детальное поведение функции;
- ***EP*** – количество расширенных протоколов, кодирующих поведение вызываемых функций;
- ***CP*** – количество протоколов-коннекторов, необходимых для моделирования вызовов функций.

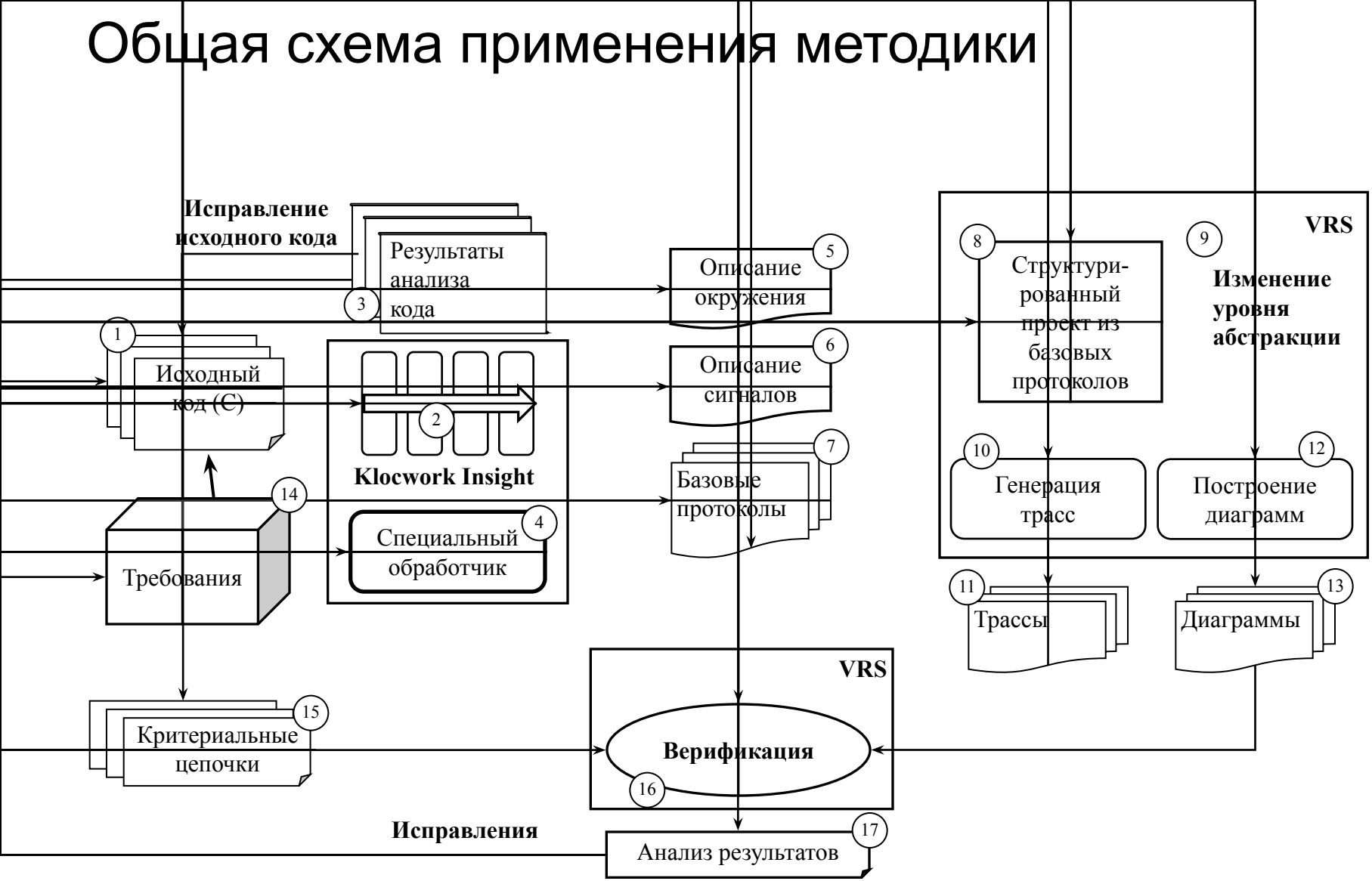
---

$$BP = (LOC + 1) + i + e + f + s + 2 \cdot w + F$$

$$EP = \left( F^{def} + F^{undef} \right) \qquad CP = 2 \cdot \left( F^{def} \right)$$

- ***LOC*** – количество строк кода функции, каждая из которых содержит хотя бы один оператор;
- ***i, e, f, s, w, F*** – количество операторов if, else, for, switch, while и вызовов функций в коде функции соответственно.

# Общая схема применения методики



# Проекты пилотирования и применения методики

Пилотирование и применение разработанного комплекса методик и программных средств проведено в следующих 4 проектах:

- **Учебный проект**. Применение методики к исходному коду приложения с целью проверки всех разработанных методик и программных средств (40 BPs).

- **Проект автомобильного радио (CarRadio)**. Применение методики структурирования базовых протоколов для получения проекта, структура которого позволяет работать с моделью покомпонентно и на разных уровнях абстракции (70 BPs).

- **Проект анализатора A-деревьев**. Применение методики к исходному коду реализованного обработчика с целью проверки корректности его реализации (8000 BPs).

- **Приложение для тестирования мобильного телефона**. Применение методики к исходному коду приложения для мобильного телефона с целью верификации реализованного приложения (70000 BPs).

# Анализ результатов применения

## Зависимость трудозатрат от размеров модели (аппроксимация на основе пилотирования)

**Тип А**




**Тип Б**




$C_A^A = 0{,}1$ чел.-часа

$C_P^A = 0{,}16 * (кол-во\ бп)$ чел.-часа

$C_A^A$ - трудозатраты автоматиз. подхода

$C_P^A$ - трудозатраты ручного подхода

$C_P^Б = 0{,}1 + 0{,}08 * (кол-во\ бп)$ чел.-часа

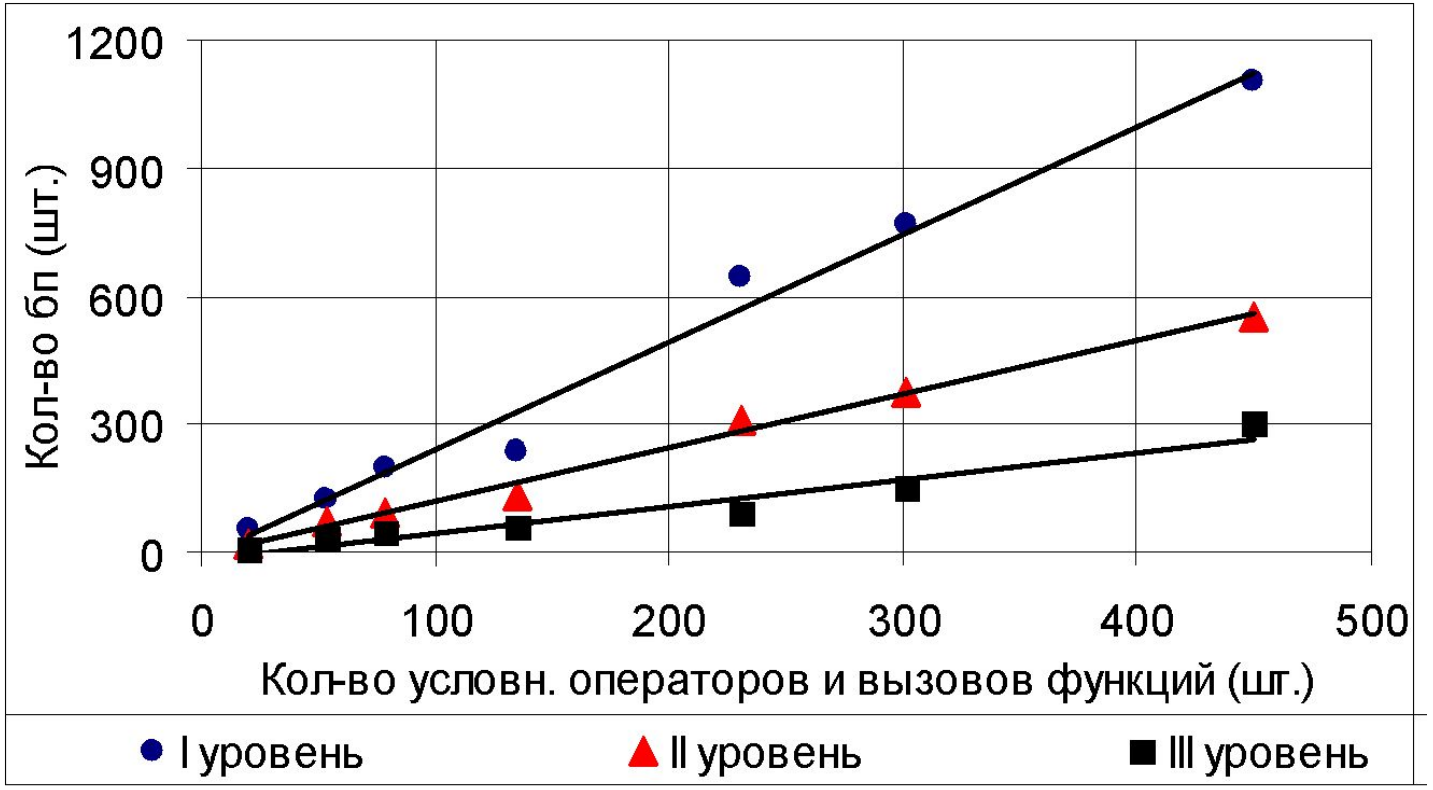
$C_P^Б = 0{,}25 * (кол-во\ бп)$ чел.-часа

$C_A^Б$ - трудозатраты автоматиз. подхода

$C_P^Б$ - трудозатраты ручного подхода

# Анализ результатов применения

Зависимость размеров моделей от уровня абстракции (аппроксимация на основе пилотирования)



# Заключение

- На основе теории агентов и сред предложена модель поведения С-приложений в виде структурированного множества базовых протоколов, пригодная для статического и визуального анализа поведенческих и структурных свойств в среде инсерционного программирования.
- Разработана методика структуризации представления модели, обеспечивающая свойство декомпозиции модели на структурные элементы и их независимый анализ на заданном уровне абстракции.
- Разработана методика использования расширенных протоколов и протоколов-коннекторов для спецификации и моделирования вызовов функций и других фрагментов исходного кода, обеспечивающая сокращение размеров модели и достижение различной степени ее детализации
- Создана программная реализация разработанных методик формализации исходного кода С-приложений, обеспечивающая автоматизацию построения поведенческих моделей.
- Оценка эффективности разработанных методик и ПО проведена в 4-х программных проектах различной сложности и позволила установить минимум трехкратное преимущество по трудозатратам автоматизированного подхода перед ручным.
- Анализ результатов позволил получить оценки применения методики в промышленных проектах.

# На защиту выносятся

- модель поведения приложений, реализованных на языке C, представляемая структурированным множеством базовых протоколов. Модель является пригодной для статического и визуального анализа ее поведенческих и структурных свойств в среде инсерционного программирования;

- методика структуризации представления модели, позволяющая проводить ее докомпозицию на структурные элементы и их независимый анализ, что обеспечивает возможность работы с крупными моделями промышленных систем;

- методика использования расширенных протоколов для формализации отдельных фрагментов исходного кода, обеспечивающая сокращение размеров модели и предоставляющая возможность достижения различной степени ее детализации;

- программные средства, обеспечивающие автоматизацию построения формальных моделей C-приложений по их исходному коду;

- проверка работоспособности предложенных методик и инструментальных средств в ряде учебных и промышленных проектов.

СПАСИБО ЗА ВНИМАНИЕ