

# **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 C-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 C, пригодной для статического и визуального анализа;
- создание методик формализации фрагментов исходного кода C-приложений с помощью выбранной формальной нотации;
- разработка программной реализации, позволяющей обеспечить генерацию формальных спецификаций по фрагментам исходного кода C-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

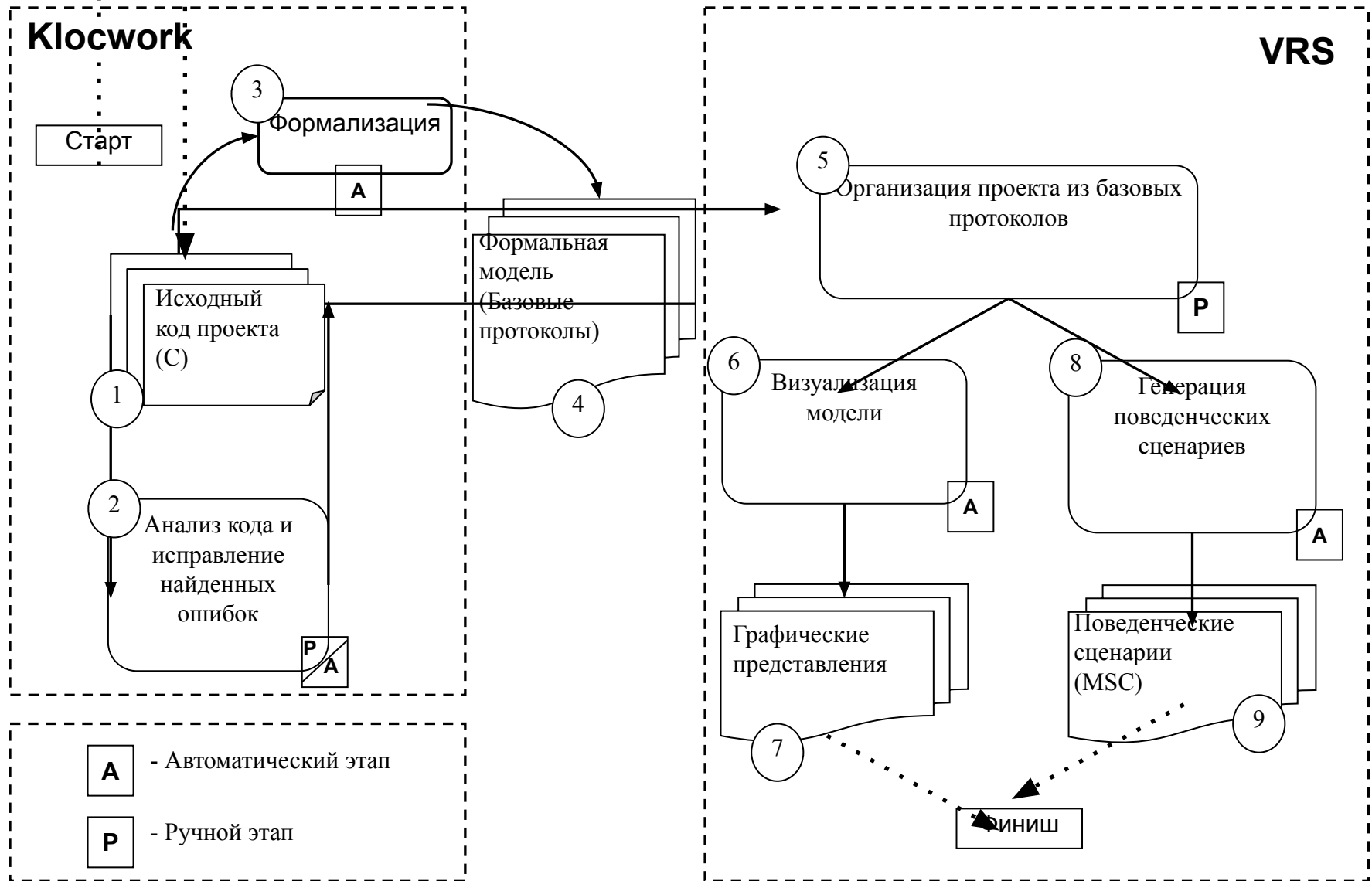
	<i>Klocwork Insight</i>	<i>Source- Navigator</i>	<i>CodeSurfer</i>	<i>CC-Rider</i>	<i>Imagix 4D</i>	<i>Cristal FLOW</i>
Целевая платформа	Win32, Unix	Win32, Unix	Win32, Unix	Win32, Unix	Win32, Unix	Win32
Поддержка языков	C, C++, C#, Java	C, C++, Java, Tcl, FORTRAN, COBOL	C, C++	C, C++	C, C++, Java	C, C++
<b>Объем поддерж. кода (MLOC)</b>	<b>&gt;5</b>	<b>0.5</b>	<b>0.5</b>	<b>1</b>	<b>&lt;1</b>	<b>1</b>
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
<b>Расширение баз. функцион. (язык прогр-я)</b>	<b>Да (C,C+)</b>	<b>Да (Tcl)</b>	<b>Да (SCHEME)</b>	<b>Да (C,C++)</b>	<b>Нет</b>	<b>Нет</b>
<b>Доступ к внутр. представл. кода</b>	<b>Да</b>	<b>Да</b>	<b>Да</b>	<b>Да</b>	<b>Нет</b>	<b>Нет</b>
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
<b>MSC</b>	CSP	Telelogic SDL TTCN, Telelogic TAU G2, VRS	+	-	-	Сценарий	-
<b>SDL</b>	CFSM	Telelogic SDL TTCN	+	+	+	Кон.-авт. диаграмма	+
<b>UML</b>	CFSM	Telelogic TAU G2	+	+	+	Кон.-авт. диаграмма	+
<b>Basic protocols</b>	ATS	VRS	+	+	+	Базовый протокол	+
<b>Promela</b>	CSP	Spin	+	-	-	Процесс	+
<b>Lotos</b>	CCS, CSP	LOTOS	+	-	-	Процесс	-
<b>Estelle</b>	CNSM	EDT	+	-	-	Кон.-авт. диаграмма	+
<b>VDM-SL</b>	LPF	VDM	+	-	-	Функция	-
<b>RSL</b>	VDM-SL, CSP	RAISE	+	-	-	Процесс	-

# Концепция предлагаемого подхода

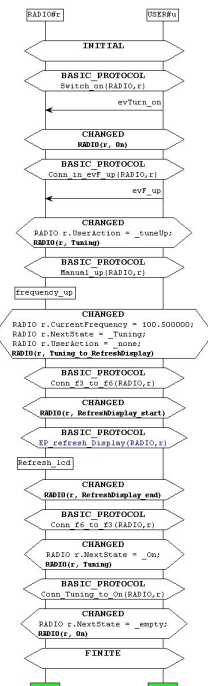
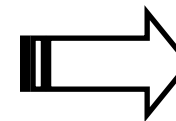
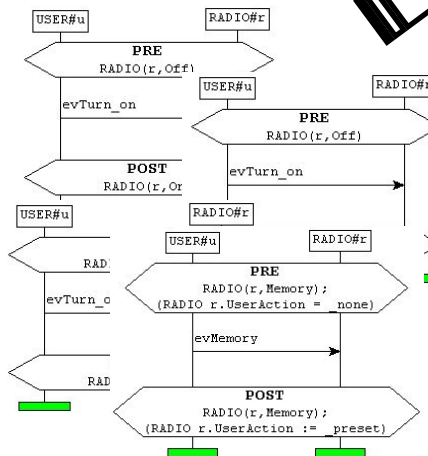
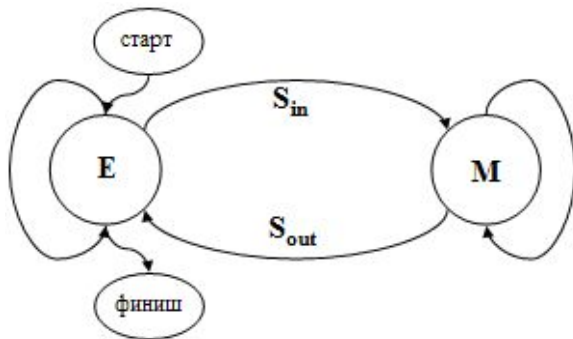
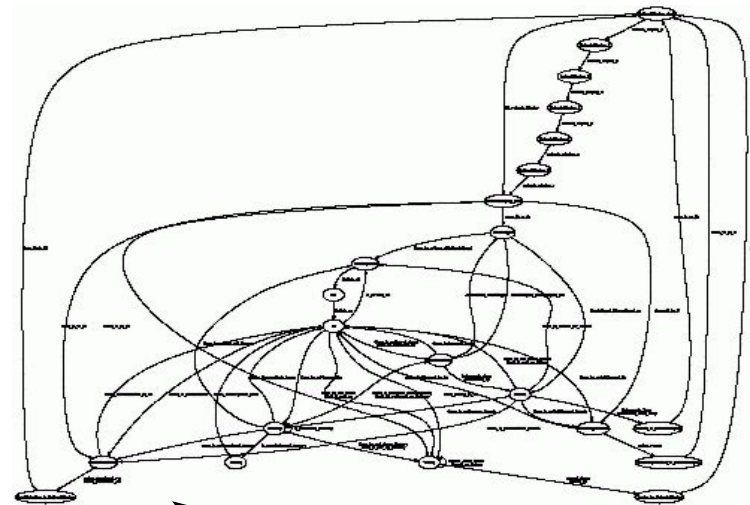


# Аппарат описания модели поведения программной системы

## Атрибутная транзиторная система

$\langle S, A, T, L, I \rangle$

- $S$  – множество состояний;
- $A$  – множество действий;
- $T$  – множество размеченных переходов ( $s \xrightarrow{a} s'$ ) и неразмеченных (скрытых) переходов ( $s \rightarrow s'$ )
- $L$  – множество атрибутивных разметок;
- $l: S \rightarrow L$  – частично определенная функция разметки состояний.



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

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



# Динамические аспекты модели поведения

## 1. Исходный код

```

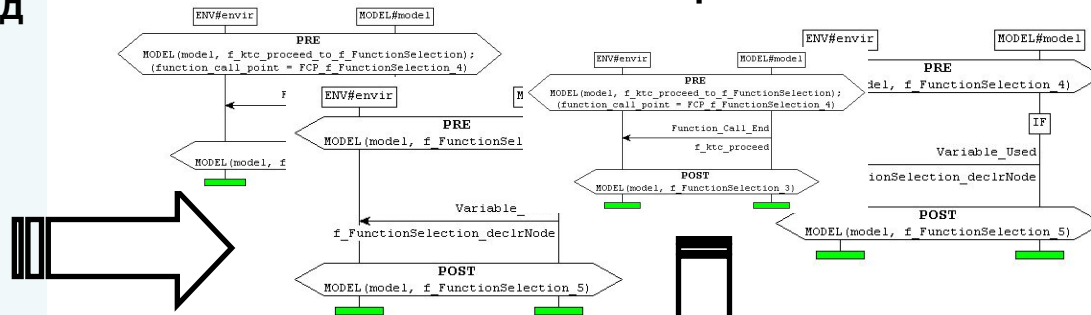
int pid;
int replyEvent;
char *replyMessage;
void *replyPointer;
int len;
int notification = 0; /* not notification by default */
char *rpcmode = ""; /* no extra text by default */

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);
}

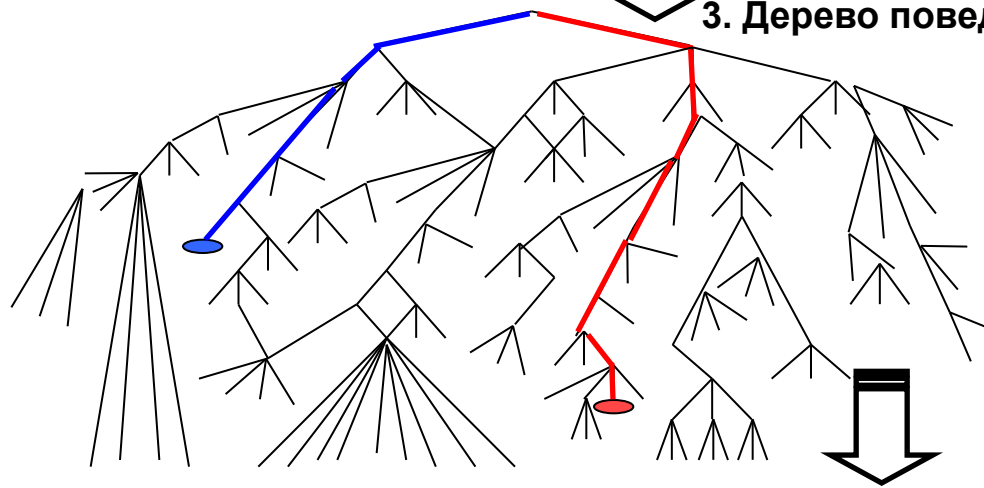
if (argc > 3 && !strcmp (argv[argc-1], "-notification" )) {
notification = 1;
rpcmode = "NOTIFICATION";
argc--;
argv[argc] = 0; /* hide this flag to avoid later confusion
*/
}

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

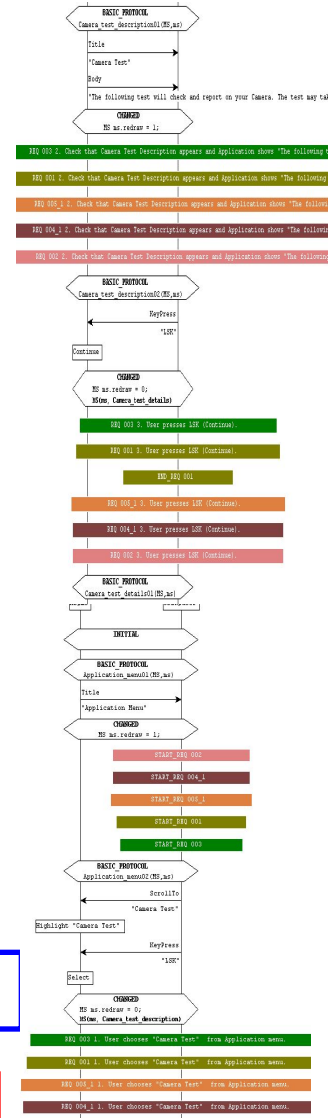
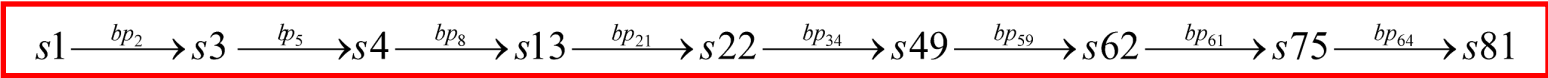
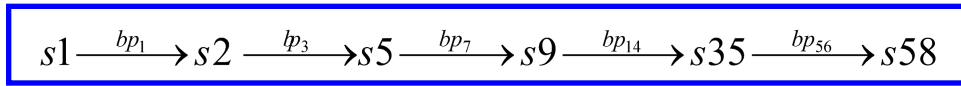
## 2. Базовые протоколы



## 3. Дерево поведения

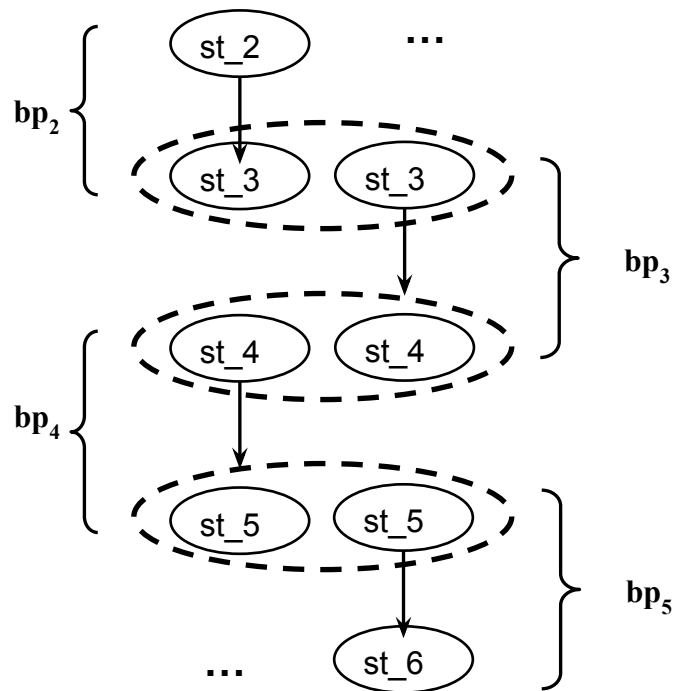


## 4. Поведенческие сценарии



# Методика 1: сохранение потока управления программы

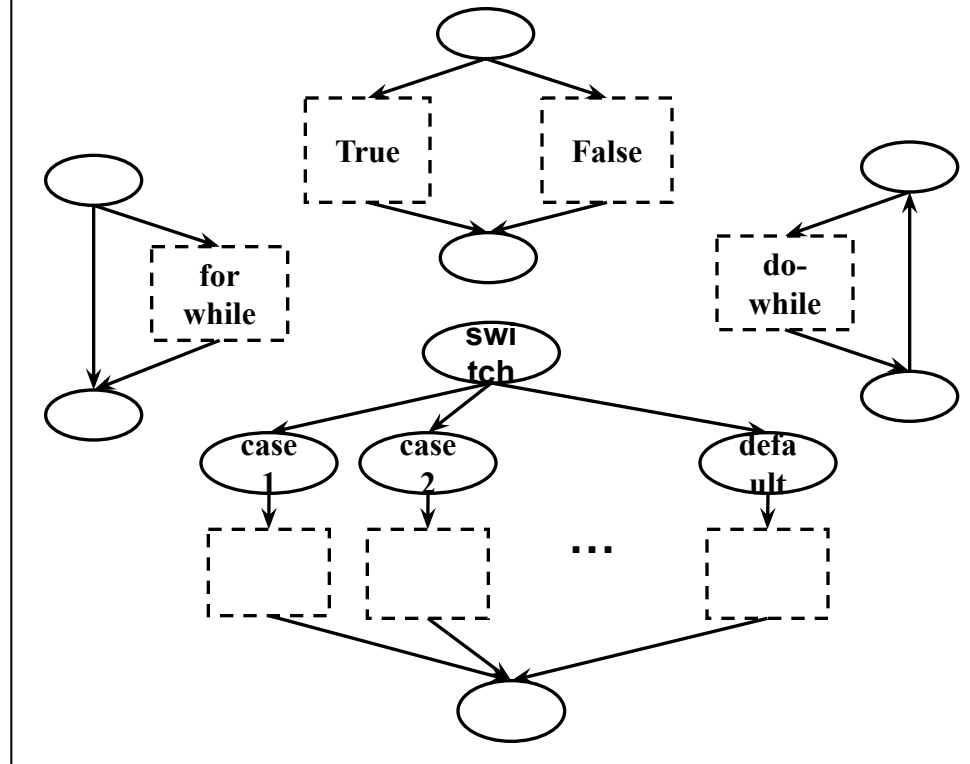
Связь базовых протоколов по состояниям агента-приложения



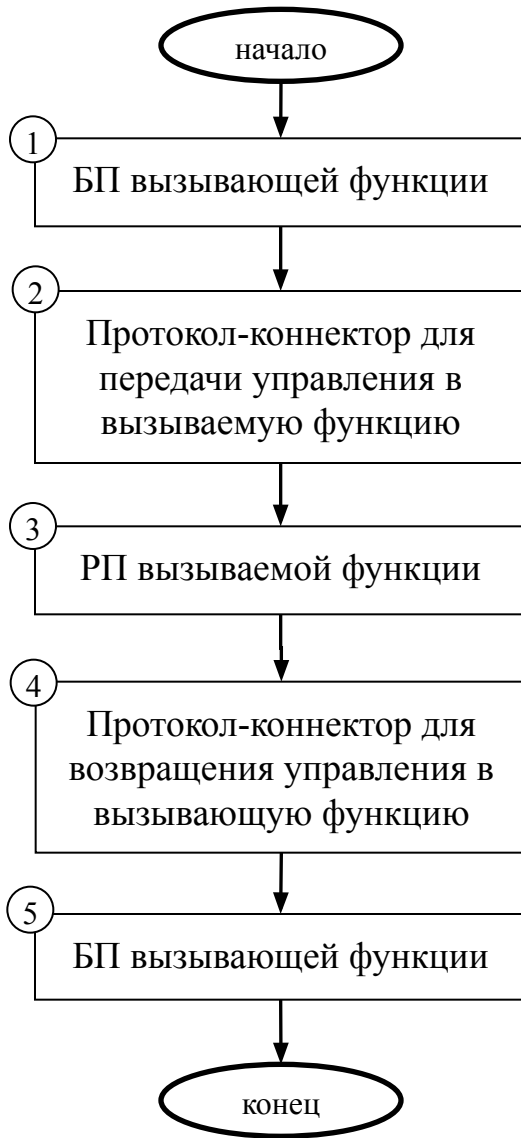
$\dots st\_2 \xrightarrow{bp_2} st\_3 \xrightarrow{bp_3} st\_4 \xrightarrow{bp_4} st\_5 \xrightarrow{bp_5} st\_6 \dots$

st\_2, st\_3, st\_4, st\_5, st\_6 – состояния агента-приложения;  
bp\_2, bp\_3, bp\_4, bp\_5 – базовые протоколы.

Фрагменты систем переходов для нелинейных фрагментов кода

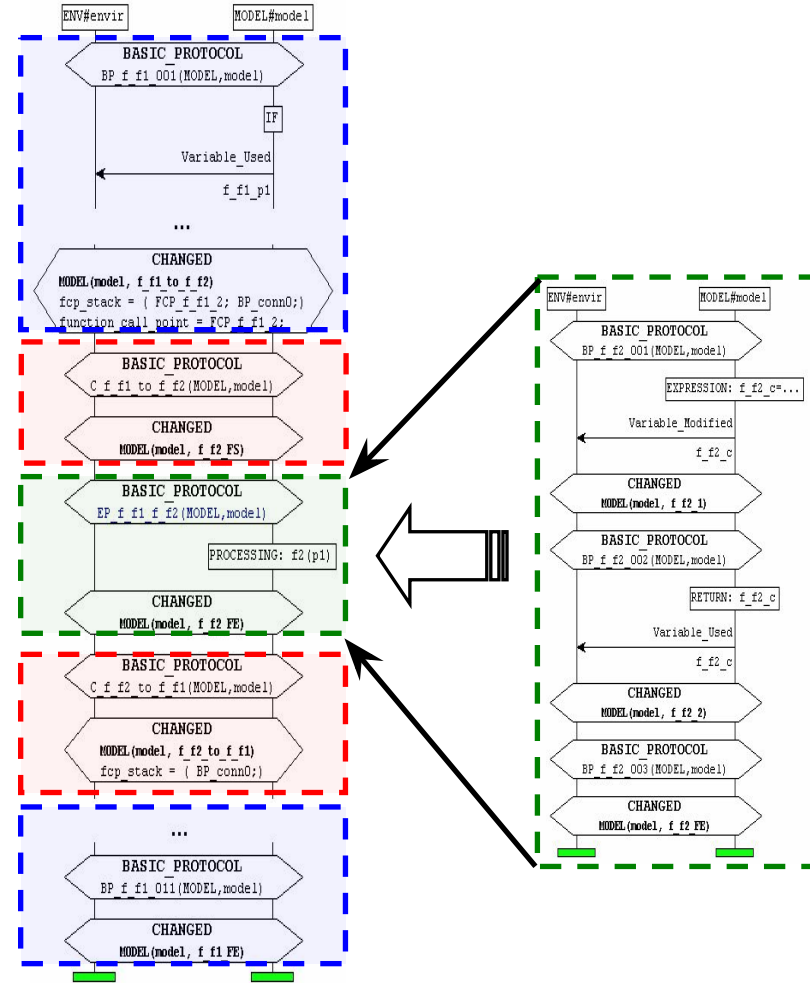


# Методика 2: формализация вызовов функции (1)



$$\begin{aligned}
 &1) \\
 &bp_{n-1} = (a_{n-1}, a_n) \in A \\
 &2) cp_1 = (a_n, b_1) \\
 &(a_n) \in A \\
 &(b_1) \in B \\
 &3) ep = (b_1, b_m) \\
 &(b_1, b_m) \in B \\
 &4) \\
 &cp_2 = (b_m, a_{n+1}) \in B \\
 &(a_{n+1}) \in A \\
 &5) \\
 &bp_{n+1} = (a_{n+1}, a_{n+2}) \in A
 \end{aligned}$$

A – множество состояний вызывающей функции  
 B – множество состояний вызываемой функции

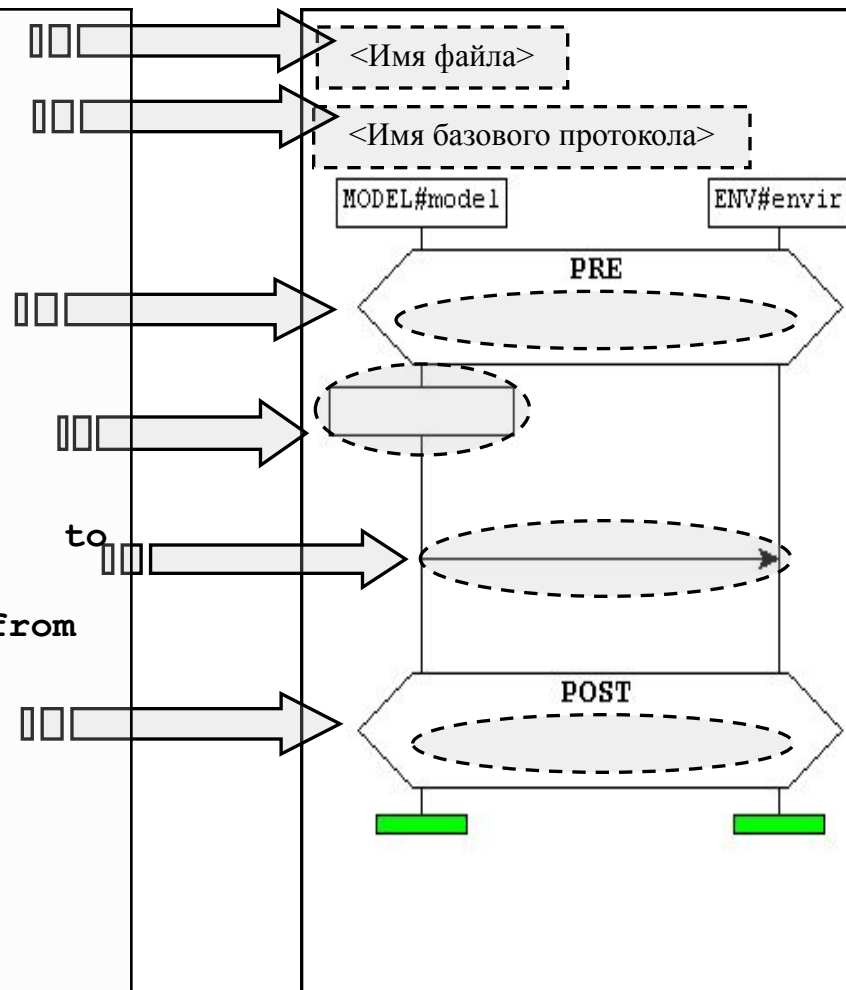


# Методика 3: построение базовых протоколов

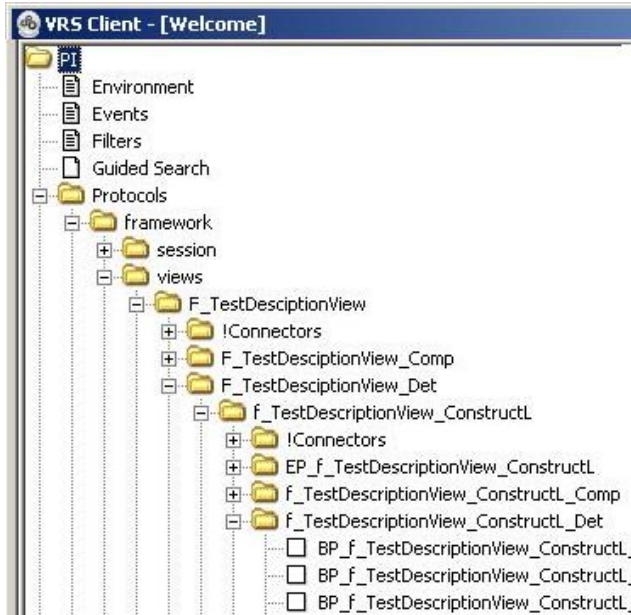
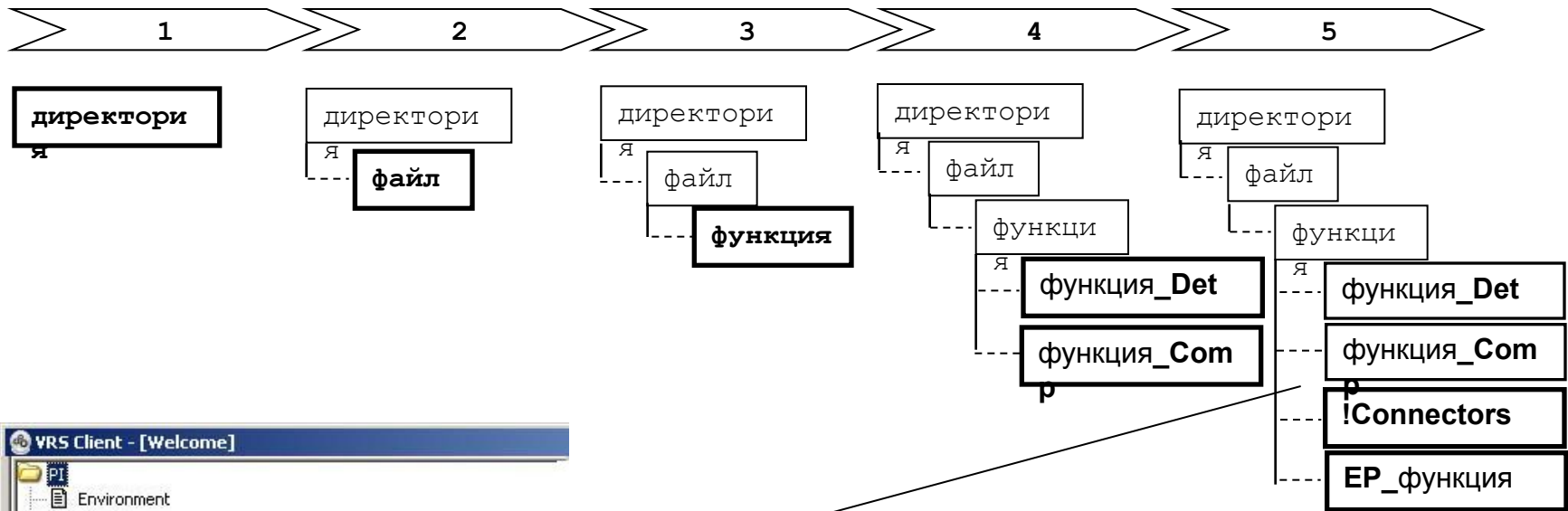
## Текстовое представление шаблона MSC диаграммы

```
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#envir;  
ENV#envir: 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



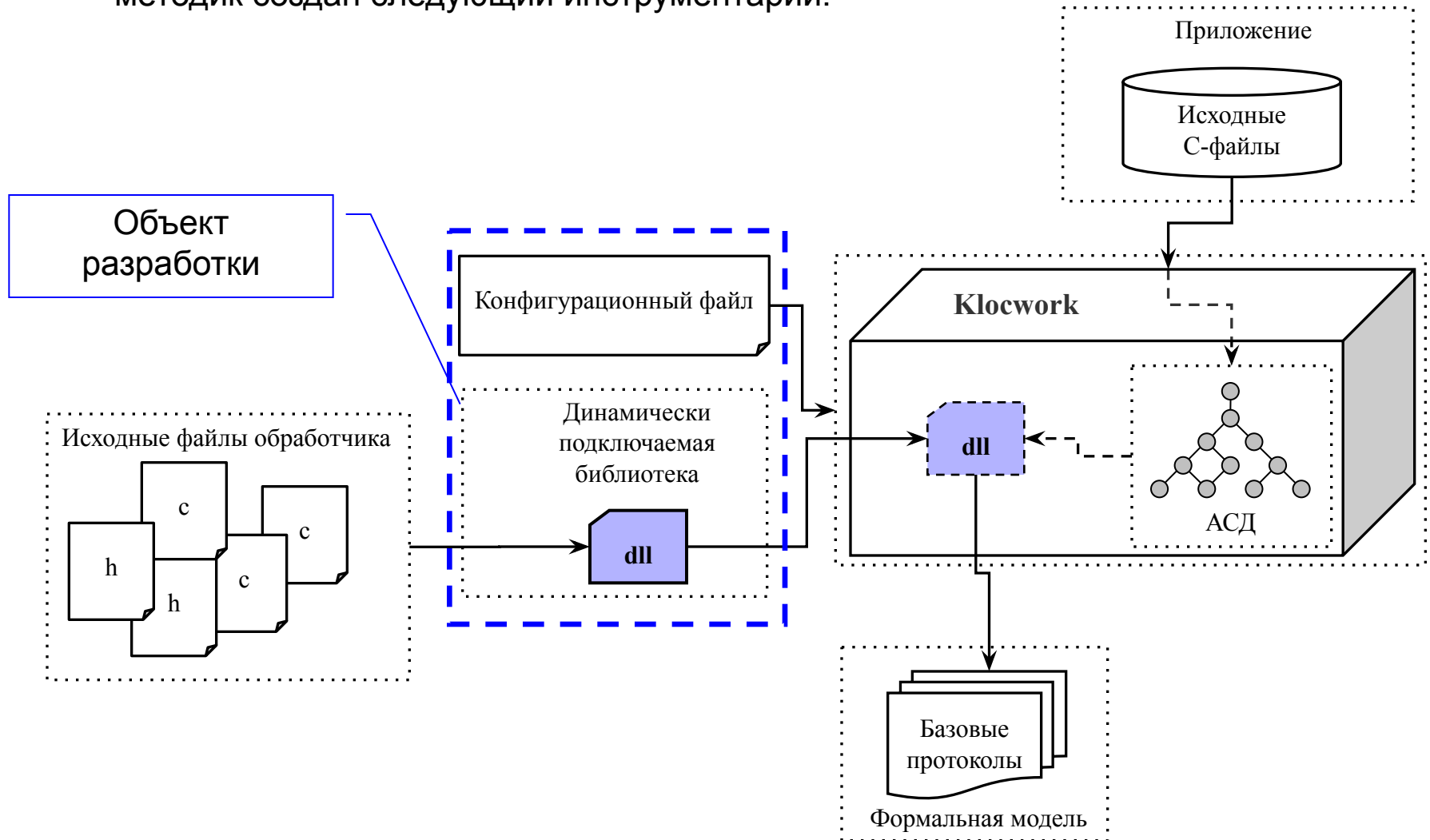
# Методика 4: структурирование базовых протоколов



- **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

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

- ***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.
- 

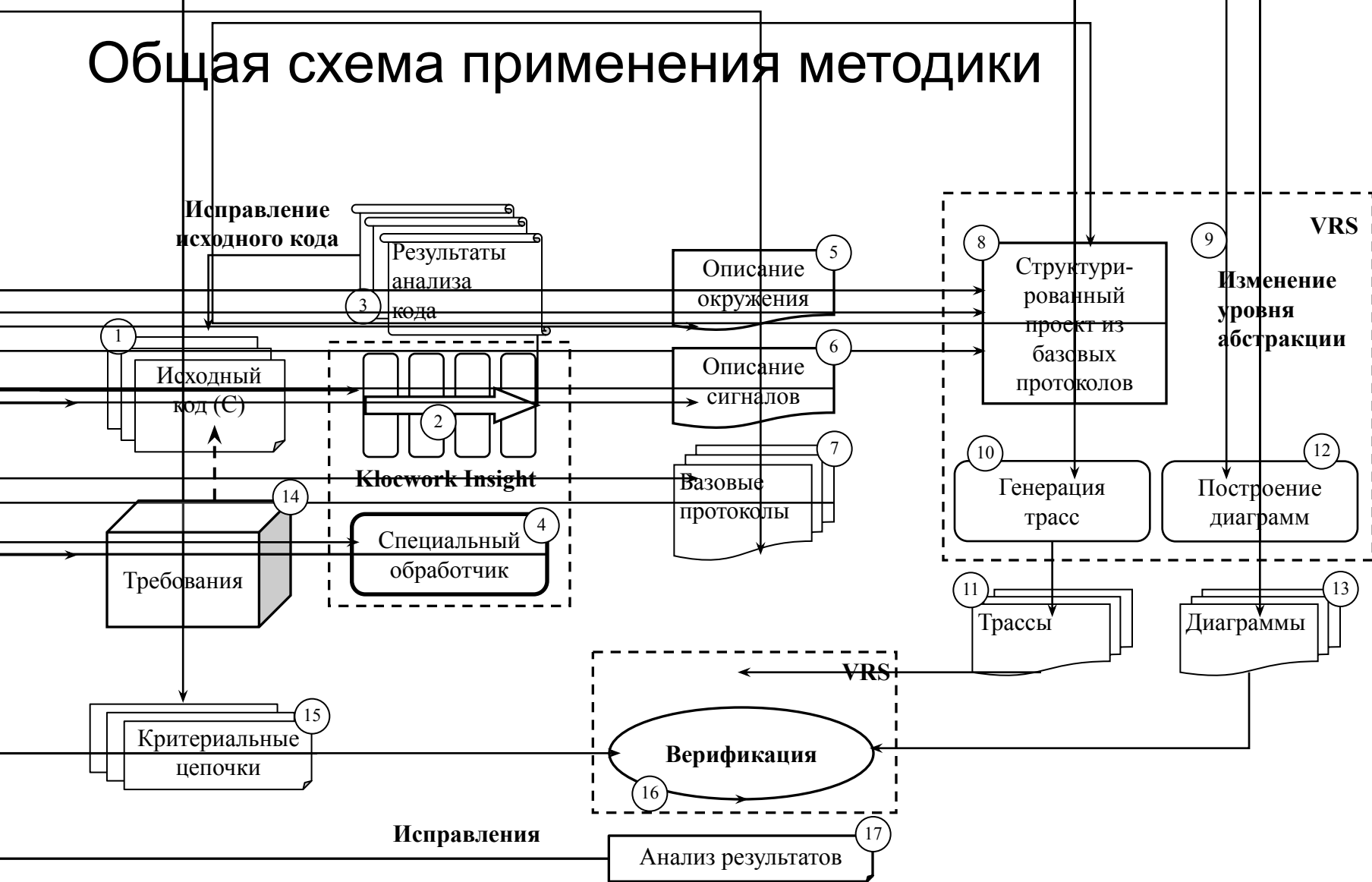
$$BP = (LOC + 1) + i + e + f + s + 2 \cdot w + F$$

$$EP = (F^{def} + F^{undef})$$

$$CP = 2 \cdot (F^{def})$$

- ***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

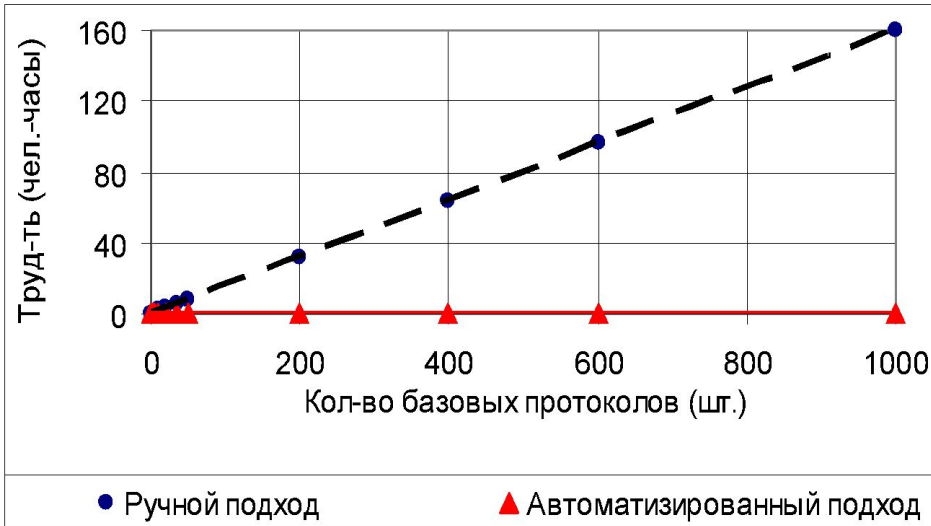
Пилотирование и применение разработанного комплекса методик и программных средств проведено в следующих 4 проектах:

- **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 А-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).

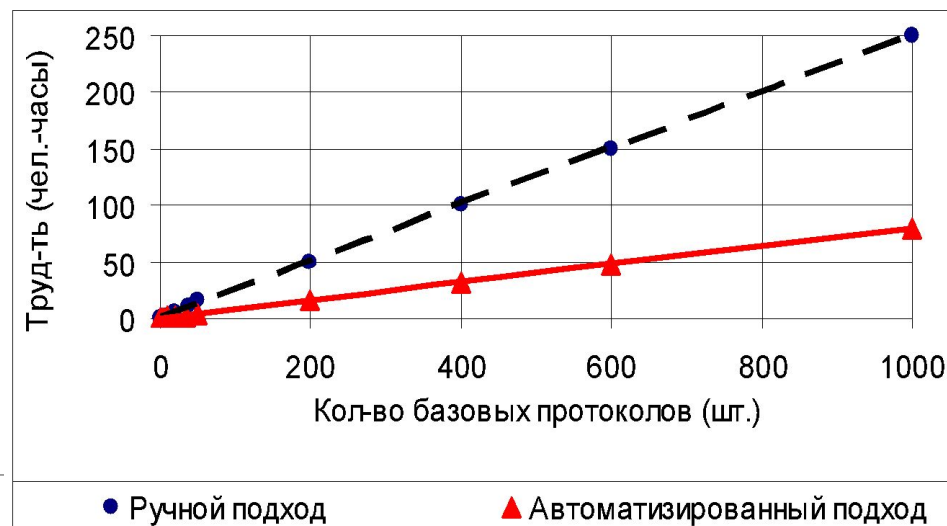
# Анализ результатов применения

Зависимость трудозатрат от размеров модели  
(аппроксимация на основе пилотирования)

Тип А



Тип Б



$$C_A^A = 0,1 \text{ чел.-часа}$$

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

$C_A^A$  - трудозатраты автоматиз. подхода

$C_P^A$  - трудозатраты ручного подхода

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

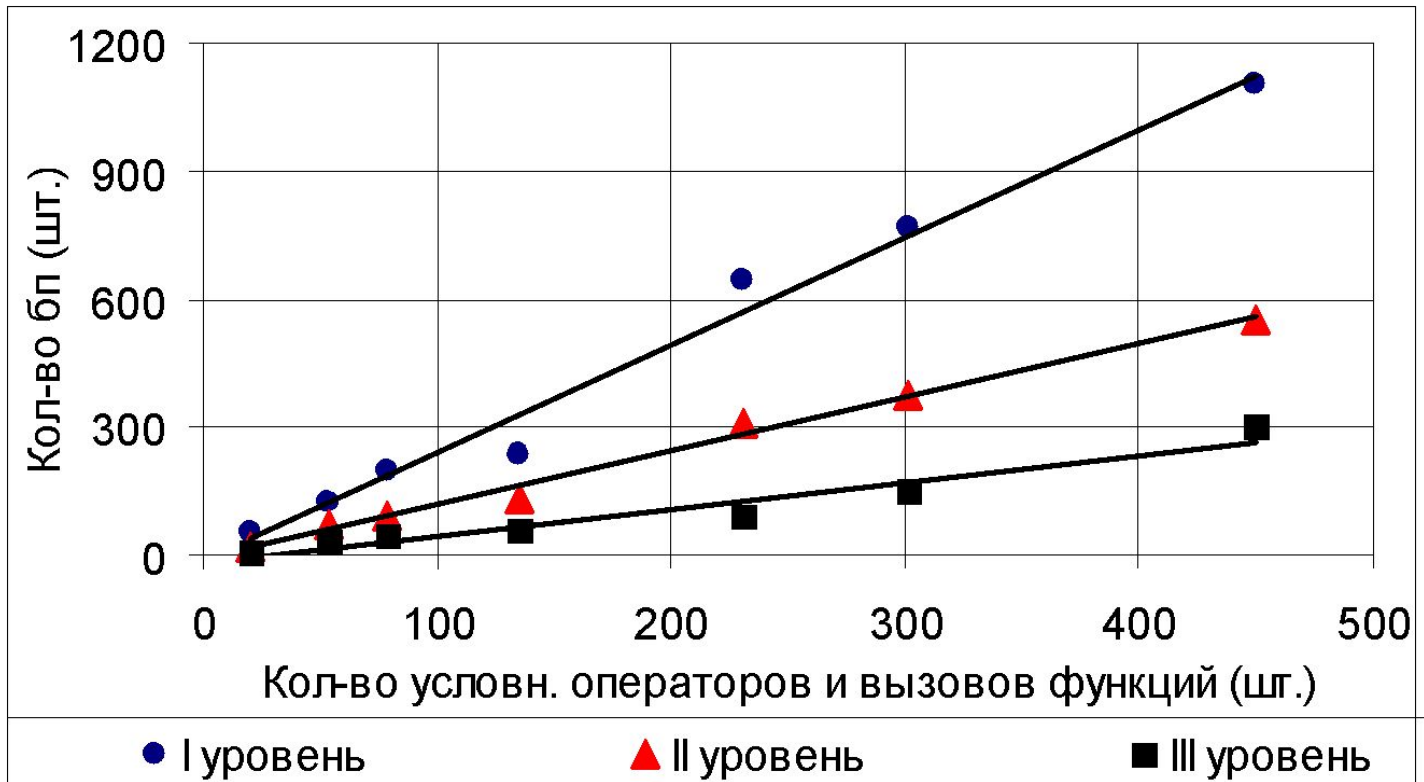
$$C_A^B = 0,25 * (\text{кол} - \text{во бп}) \text{ чел.-часа}$$

$C_A^B$  - трудозатраты автоматиз. подхода

$C_P^B$  - 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 С, 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 С-приложений по их исходному коду;
- проверка работоспособности предложенных методик и инструментальных средств в ряде учебных и промышленных проектов.

**СПАСИБО ЗА ВНИМАНИЕ**