Кандидатская диссертация

**Курдюков Николай Станиславович**

# МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ИНТЕЛЛЕКТУАЛЬНЫХ СЕРВИС-ОРИЕНТИРОВАННЫХ СИСТЕМ НА ОСНОВЕ ИСПОЛЬЗОВАНИЯ ЯЗЫКОВ ДЕСКРИПТИВНОЙ ЛОГИКИ

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

Научный руководитель: доктор технических наук
Каширин Игорь Юрьевич

Рязань 2014

## На защиту выносятся следующие новые научные результаты

1. Формализм для описания и анализа концептуальной модели SOA-системы, ориентированный на онтологический подход и позволяющий адекватно описывать взаимодействие web-сервисов.

2. Новые алгоритмы автоматизированного построения интерфейсов web-сервисов для SOA-архитектуры, дающие возможность автоматизировано строить интерфейсы web-сервисов.

3. Программная реализация SOA-системы, предназначенная для использования в основе систем автоматизированного построения интерфейсов взаимодействия web-сервисов.

4. Структура онтологии, необходимой для работы системы, имеющая OWL-разметку стандарта Semantic web.

# ОБЛАСТЬ ИССЛЕДОВАНИЯ

На данный момент существует множество систем для создания web-приложений с SOA архитектурой: IBM Lotus Notes, Microsoft .NET, Oracle SOA Suite. В тоже время не существует эффективных систем, способных автоматически строить интерфейсы и превращать популярные сайты прежнего поколения в элементы композитных служб или сервисных кластеров.

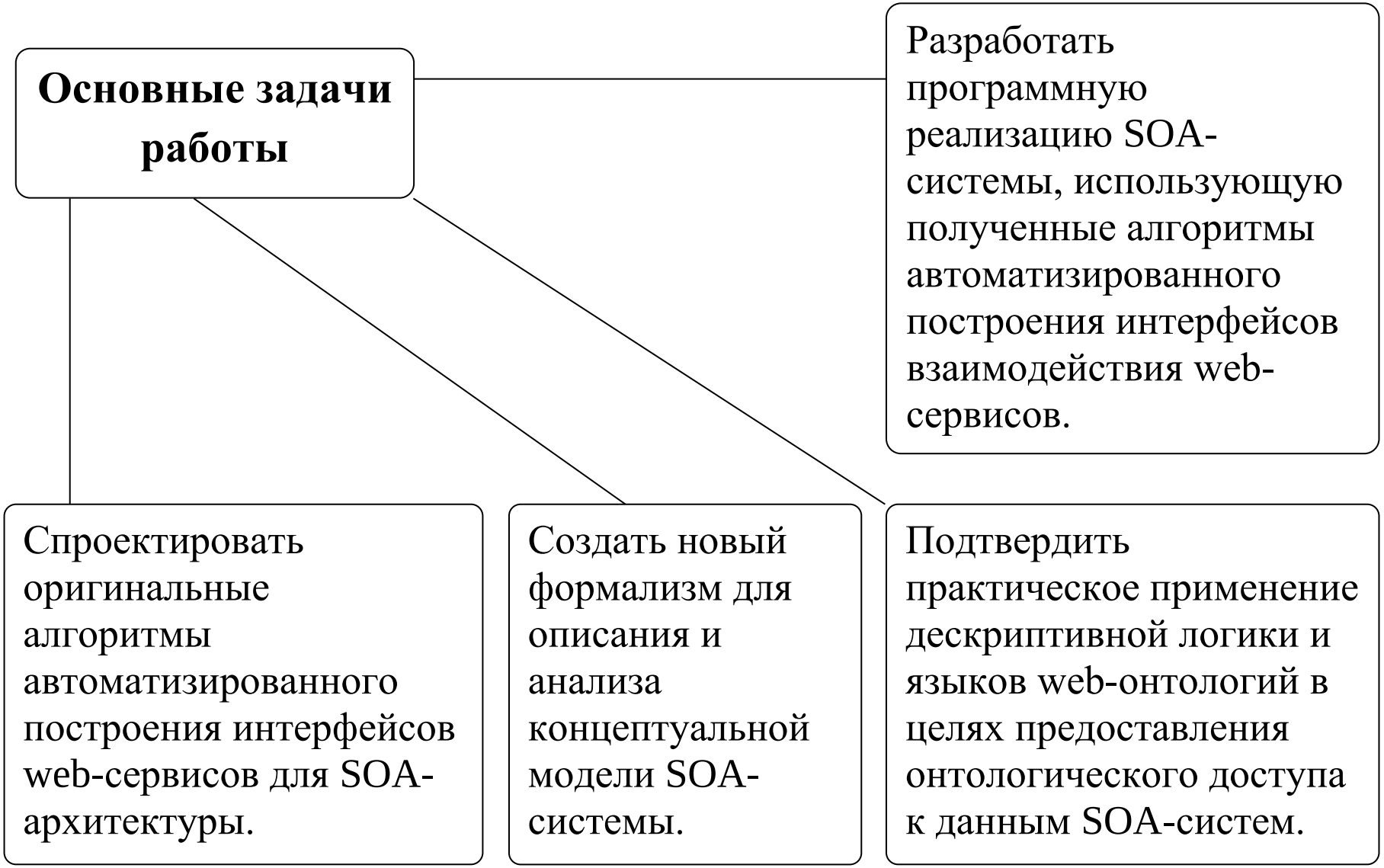
Необходимость таких систем обусловливается проблемами реализации EAI (Enterprise Application Integration - интеграции корпоративных приложений).

Поддержке SOA-систем, активно работающих с сетью Internet, мешает фактор постоянного развития всемирной паутины. При этом постоянное изменение архитектуры и разработка заново уже существующих интерфейсов требует непрерывной работы специалистов.

OBDA – это доступ к данным через высокоуровневый интерфейс онтологии. Центральным процессом основанного на онтологиях доступа к данным является представление базы данных на языке дескриптивной логики(ДЛ). Такой подход позволяет использовать преимущества онтологий и ДЛ.

# НАПРАВЛЕНИЕ ИССЛЕДОВАНИЙ

**Цель диссертации:** создание формализма описания взаимодействия сервисов и алгоритмов построения интерфейсов web-сервисов для создания эффективной SOA-системы с применением парадигмы основанного на онтологиях доступа к данным.



# СТРУКТУРА ДИССЕРТАЦИИ

**Работа состоит из введения, четырех глав, заключения, приложения и списка литературы**

**Глава 1**

Содержит информацию о состоянии и тенденциях развития SOA-технологий и OBDA-технологий.

**Глава 2**

Формулируются основные задачи, разрабатывается формализм для описания SOA-систем в рамках технологии дескриптивной логики.

**Глава 3**

Предлагаются алгоритмы и технологические решения к реализации основанных на онтологиях SOA-систем.

**Глава 4**

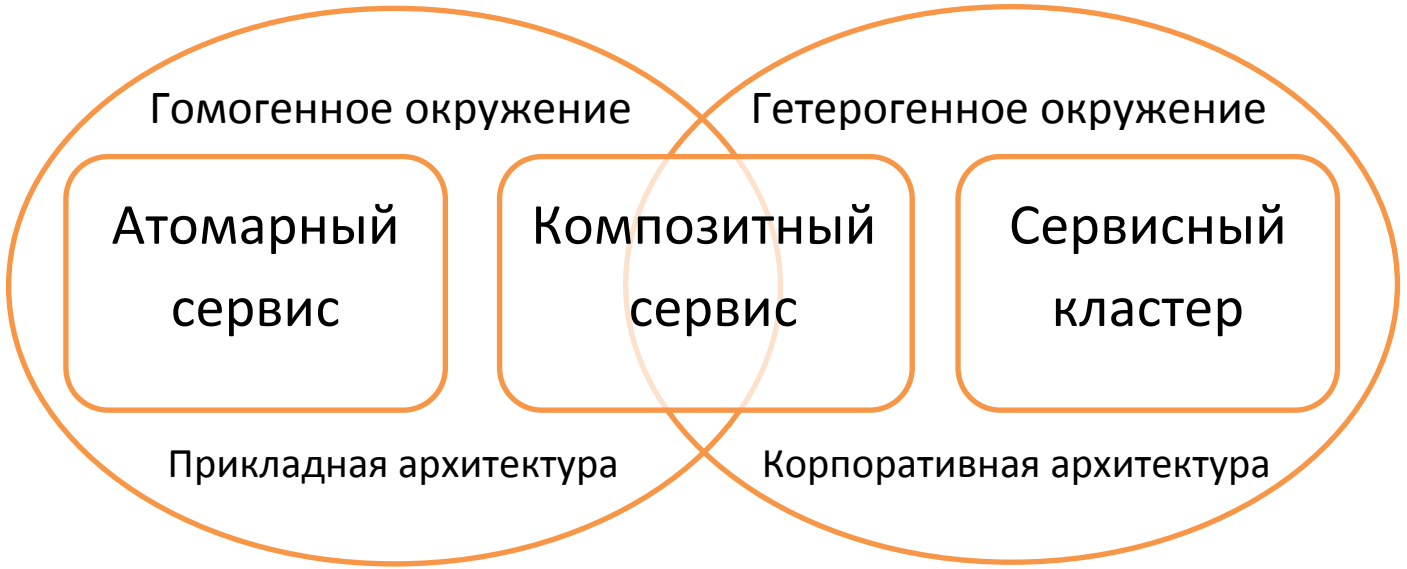
Описывается структура и возможности основанной на онтологиях SOA-системы SIRSystem v1.0

# ГЛАВА 1. СОВРЕМЕННЫЕ ПОДХОДЫ К ПРОЕКТИРОВАНИЮ SOA-СИСТЕМ

Центральным понятием SOA-архитектуры является сервис(service – услуга).

**Термин сочетает в себе следующие взаимосвязанные идеи:**

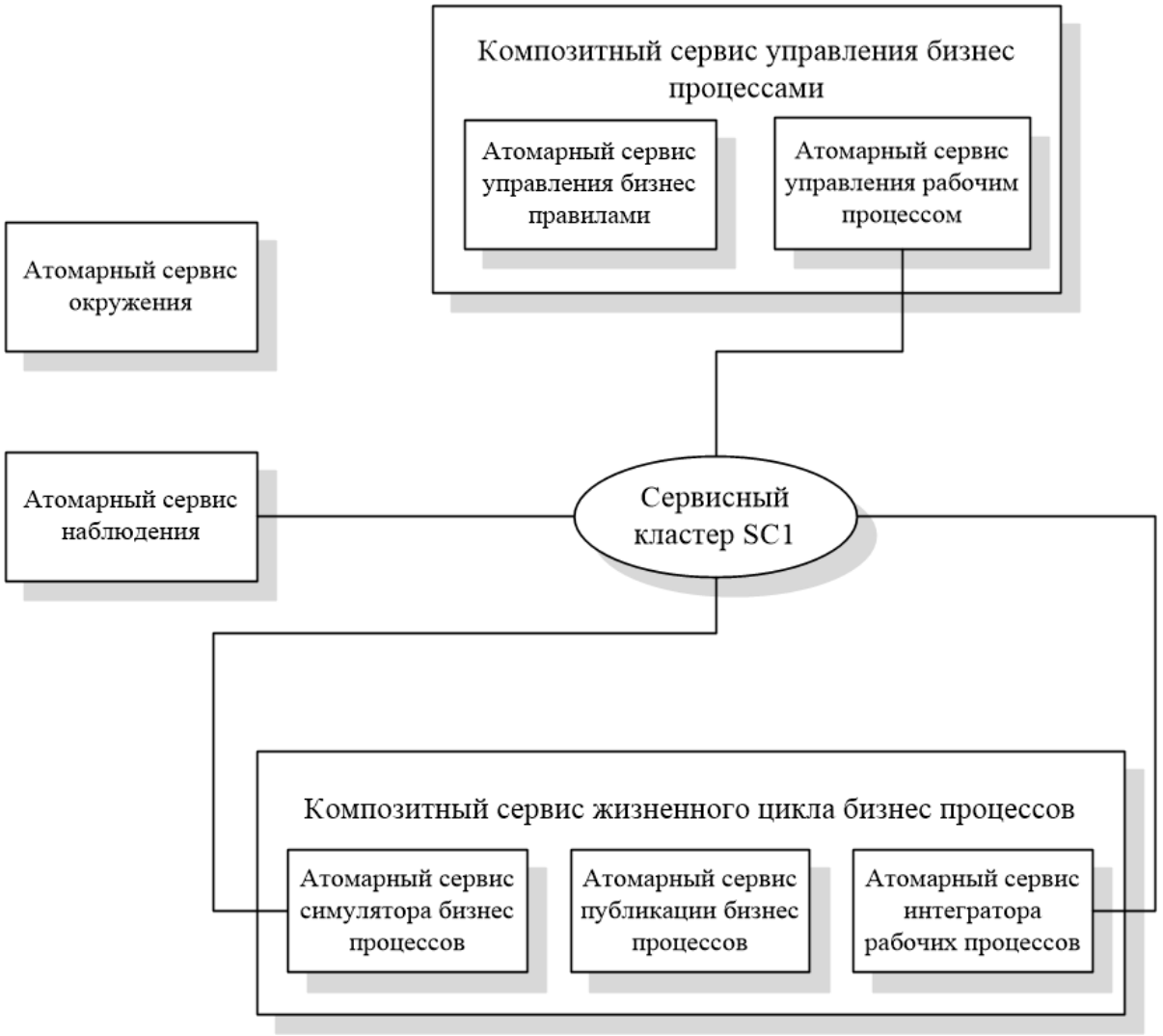
- возможность выполнять работу для клиентов;
- четкая формализация работ, предоставляемых клиентам;
- предложение выполнить работу для клиента.



# Характеристики машин вывода

| Наименование машины вывода | Формат данных | Алгоритм логического анализа | Поддерживаемые логики | Язык програм-мирования | Интеграция |
|---|---|---|---|---|---|
| Hermit | OWL 2 | hypertableau algorithm | SROIQ | Java | редактор онтологий Protégé |
| Pellet | OWL 2 | tableau algorithm | SROIQ | Java | редактор онтологий Protégé |
| FaCT++ | OWL 2 | tableau algorithm | SROIQ | C++ | редактор онтологий Protégé |
| RacerPro | OWL 1 | tableau algorithm | SHIQ | LISP | редактор онтологий Protégé |

# ГЛАВА 2. ОНТОЛОГИИ И ДЕСКРИПТИВНАЯ ЛОГИКА ДЛЯ СИСТЕМЫ АВТОМАТИЗИРОВАННОГО ПОСТРОЕНИЯ ИНТЕРФЕЙСОВ WEB-СЕРВИСОВ




Пример концептуальной модели взаимодействия сервисов

$$i_{SOA} = <C_s, Q_s, S_s>,$$

где $C_s$ – переменная, содержащая адрес web-сервиса, запрашивающего информацию (адрес клиентской части); $Q_s$ – множество запрашиваемых информационных ресурсов; $S_s$ - переменная, содержащая адрес web-сервиса запрашиваемого информацию (адрес серверной части).

$$ServI(C_s, Q_s, S_s) –$$

соответствие интерфейсов взаимодействия web-сервисов $C_s$ и $S_s$, выражающее структуру интерфейсов, реализованных между сервисом по адресу $C_s$ и сервисом по адресу $S_s$ и передающая информация $Q_s$ между ними.

**Соответствие** $\varphi_1$, является отображением области определения $ServIRend(C_s, Q_s, S_s)$ в область значений:

$$\bigcup_{i_{SOA} \in I_{SOA}} ServI(c_i, q, s_i),$$

где $C_s$, $Q_s$, $S_s$ – более общие параметры для множественного создания сервисов; $c_i$,$q_i$,$s_i$ – частные параметры для создания сервисов; $i_{SOA}$ – элемент множества $I_{SOA}$; $I_{SOA}$ – множество интерфейсов, которые подходят по данным параметрам.

# Пример синтаксиса языков дескриптивной логики

| Символ | Операция | Примеры |
|---|---|---|
| $\top$ | истина | $\exists R.\top$ |
| $\bot$ | ложь | $\exists R.\bot$ |
| $\neg$ | отрицание | $\neg C$ |
| $\sqcup$ | объединение | $C \sqcup D$ |
| $\sqcap$ | пересечение | $C \sqcap D$ |
| $\exists$ | ограничение квантором существования | $\exists R.C$ |
| $\forall$ | ограничение квантором всеобщности | $\forall R.C$ |
| $\mid\bullet$ | типизированный концепт | $C \mid\bullet D$ |
| $\leq n$, $\geq n$ | ограничения кардинальности | $\geq n\ R.C$ |

# Синтаксис языка SOA-отношений

| Теоретико-множественное описание элементов концептуальной модели SOA-архитектуры | Эквивалентные выражения интерпретации $I$ в языке дескриптивной логики $\mathcal{ALC}$ | Синтаксис языка дескриптивной логики $\mathcal{ALC}$ |
|---|---|---|
| $b_s \in B_s$ | $b^I \in B^I$ | $B(b)$ |
| $B_{s1} \subseteq B_{s2}$ | $B_1{}^I \subseteq B_2{}^I$ | $B_1 \sqsubseteq B_2$ |
| $B_{s1} \subseteq B_{s2} \cap B_{s3}$ | $B_1{}^I \subseteq B_2{}^I \cap B_3{}^I$ | $B_1 \sqsubseteq B_2 \sqcap B_3$ |
| $B_{s1} \cup B_{s2} \subseteq B_{s3}$ | $B_1{}^I \cup B_2{}^I \subseteq B_3{}^I$ | $B_1 \sqcup B_2 \sqsubseteq B_3$ |
| $B_{s1} \cup B_{s2} = \emptyset$ | $B_1{}^I \cup B_2{}^I = \emptyset$. | $B_1 \sqcup B_2 \sqsubseteq \bot$ |
| $(b_{s1}, b_{s2}) \in R$ | $(b_1{}^I, b_2{}^I) \in R^I$ | $R(b_1, b_2)$ |

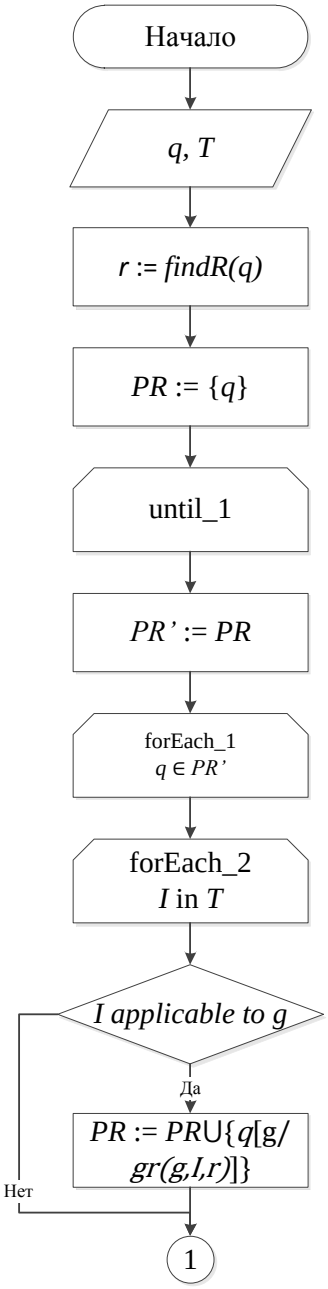
# Выражения, доступные при редукции

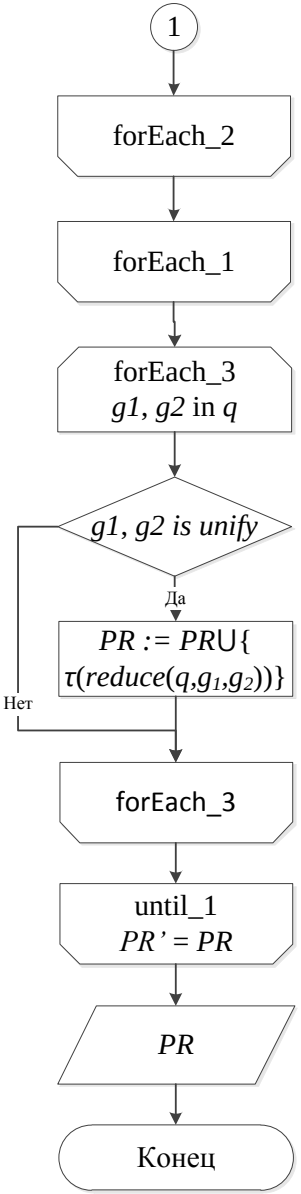
$$
\begin{array}{ccc}
B_1 \sqcup B_2 \sqsubseteq C & \equiv_T & B_1 \sqsubseteq C, B2 \sqsubseteq C; \\
B \sqsubseteq C_1 \sqcap C_2 & \equiv_T & B \sqsubseteq C_1, B \sqsubseteq C_2; \\
B_1 \equiv B_2 & \equiv_T & B_1 \sqsubseteq B_2, B_2 \sqsubseteq B_1; \\
B \sqsubseteq \exists R.\top & \equiv_T & B \sqsubseteq \exists R; \\
B \sqsubseteq \exists R.\bot & \equiv_T & B \sqsubseteq \neg\exists R; \\
B \sqsubseteq \exists R_0.C & \equiv_T & B \sqsubseteq \exists R_1, R_1 \sqsubseteq R_0, \exists R^{-}_1 \sqsubseteq C,
\end{array}
$$

где $\equiv_T$ – таксономическая эквивалентность.

# ГЛАВА 3. SIR-АЛГОРИТМЫ ДЛЯ ОНТОЛОГИЧЕСКИХ SOA-СИСТЕМ

*RewWR*



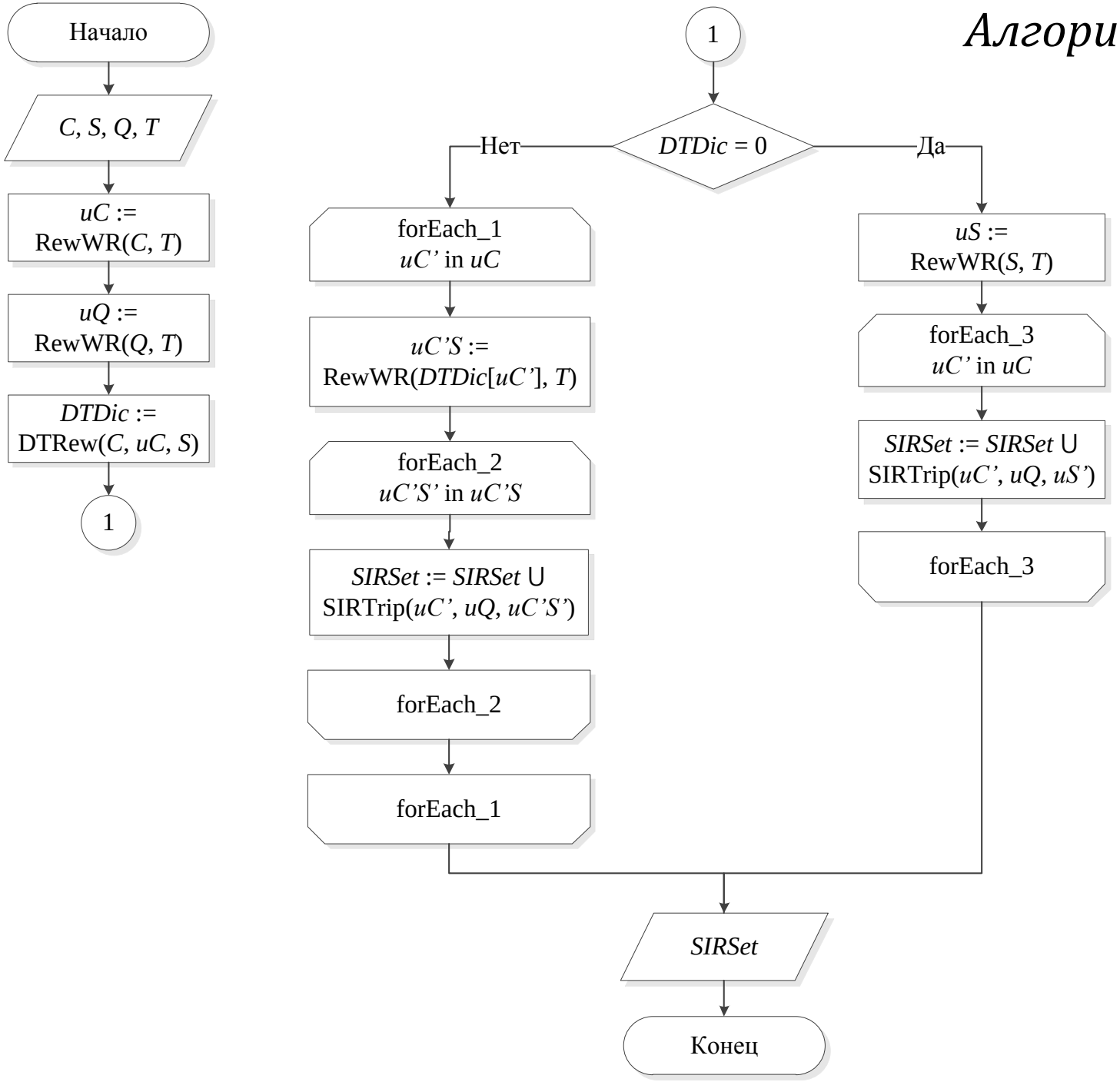
# Принцип работы алгоритма *RewWR*

1. Проверить содержание в запросе $q$ информации о ролях. Функция *findR* выдает результат «1», если в запросе находятся атомы с 2-мя аргументами, иначе результат равен «0».

2. Переформулировать атомы каждого конъюнктивного запроса $q \in PR'$ если это возможно ($I$ *applicable to* $g$) , и получить новый запрос для каждого атома переформулированной формулы. Функция $\tau$ принимает в качестве входного значения конъюнктивный запрос $q$ и возвращает новый конъюнктивный запрос, получаемый заменой каждого вхождения независимой переменной в $q$ *символом* «_».

3. Для каждой пары атомов $g_1$ и $g_2$, которая унифицируется($g_1$, $g_2$ *is unify*) и находится в теле запроса $q$, вычислить конъюнктивный запрос $q' = reduce$ ($q$, $g_1$, $g_2$). Функция *reduce* принимает в качестве входных параметров конъюнктивный запрос $q$ и два атома $g_1$ и $g_2$, находящихся в теле запроса $q$, и возвращает конъюнктивный запрос $q'$ посредством применения к $q$ унификатора для $g_1$ и $g_2$.

# *Алгоритм SIR1*



# Принцип работы алгоритма *SIR1*

За трансформацию запросов отвечает описанный ранее алгоритм *RewWR*. Алгоритм *SIR1* состоит из следующих действий.

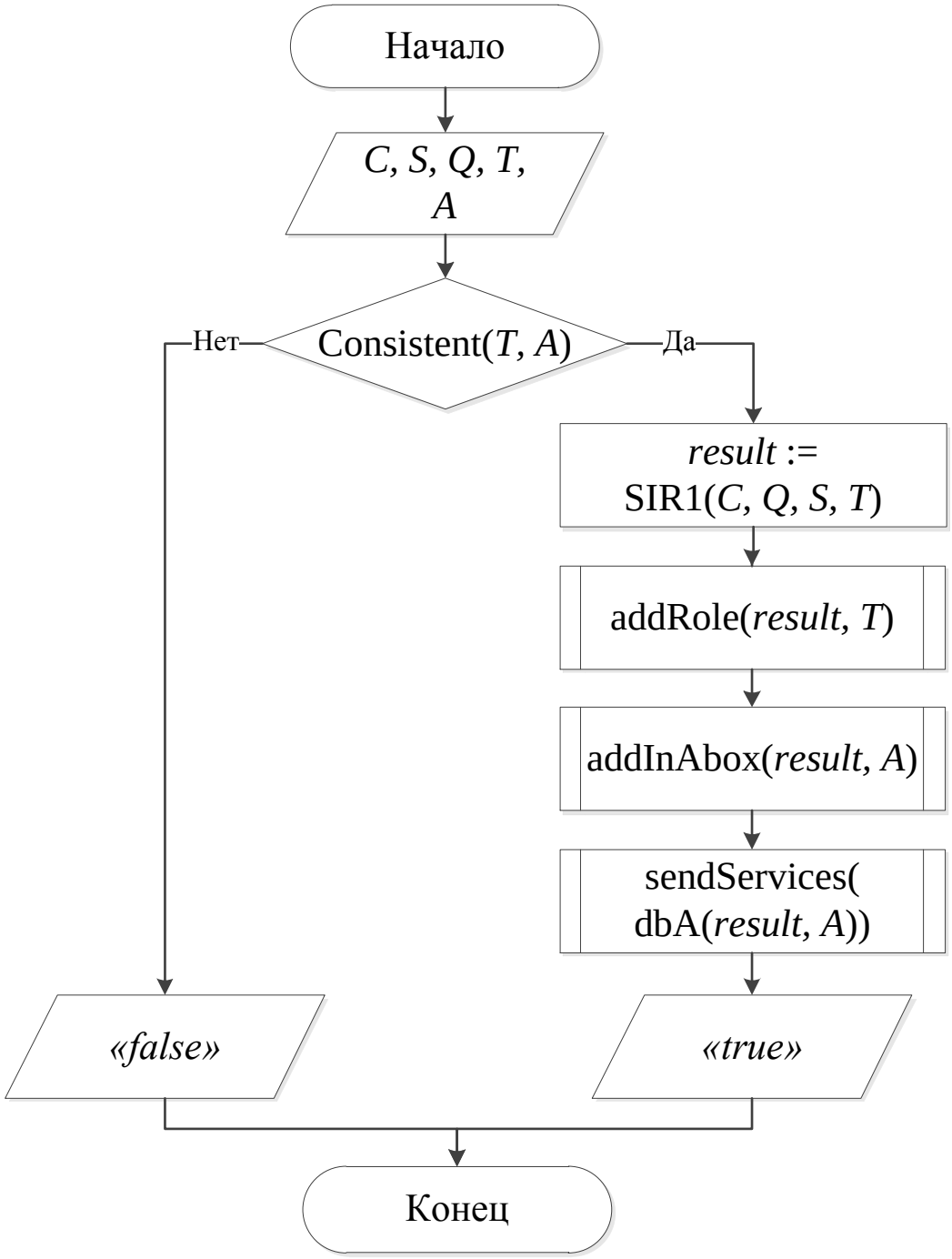
1. Посредством алгоритма трансформации запросов переписать *C* и поместить объединение запросов в множество *uC*.
2. Переписать запрос *Q* и поместить объединение запросов в множество *uQ*.

Если входные параметры не содержат зависимых элементов, то производятся следующие действия.

1. Переписать запрос *S* и поместить объединение запросов в множество *uS*.
2. Для каждого *uC'* из *uC*, создать SIR-триплет где утверждение о клиенте в ABox = *uC'*; утверждение о передаваемой информации в Abox = *uQ'* из множества *uQ*, утверждение о сервисе в ABox = *uS'* из *uS*.
3. Поместить триплет в *SIRSet*.

Если входные параметры содержат зависимые элементы, то производятся следующие действия.

1. Для обработки параметров с зависимыми элементами используется функция *DTRew*.
2. После завершения работы функции для каждого *uC'* из *uC* переписать запрос *DTDic*[*uC'*], и поместить результат в множество *uC'S*.
3. Для каждого *uC'S'* из *uC'S* создать SIR-триплет где утверждение о клиенте в ABox = *uC'*; утверждение о передаваемой информации в Abox = *uQ'* из множества *uQ*, утверждение о сервисе в ABox = *uC'S'* из множества *uC'S*.


Начало
C, S, Q, T, A
Consistent(T, A)
Нет
Да
result := SIR1(C, Q, S, T)
addRole(result, T)
addInAbox(result, A)
sendServices( dbA(result, A))
«false»
«true»
Конец

# Принцип работы алгоритма *SIRCreate*

Функция *Consistent* реализует проверку базы знаний на непротиворечивость. Если база данных оказывается противоречивой, то нет смысла проводить дальнейшие действия и алгоритм завершается.

Процедура addRole добавляет информацию о новых интерфейсах в TBox *T*. Процедура addInAbox добавляет информацию о новых интерфейсах в ABox *A*. Функция dbA выполняет запросы над ABox *A*, являющемся базой данных.

Процедура sendServices отвечает за формирование кода с запросами под БД web-сервисов и его отправку по адресам клиента и сервера на основе ответа ABox SOA-системы.

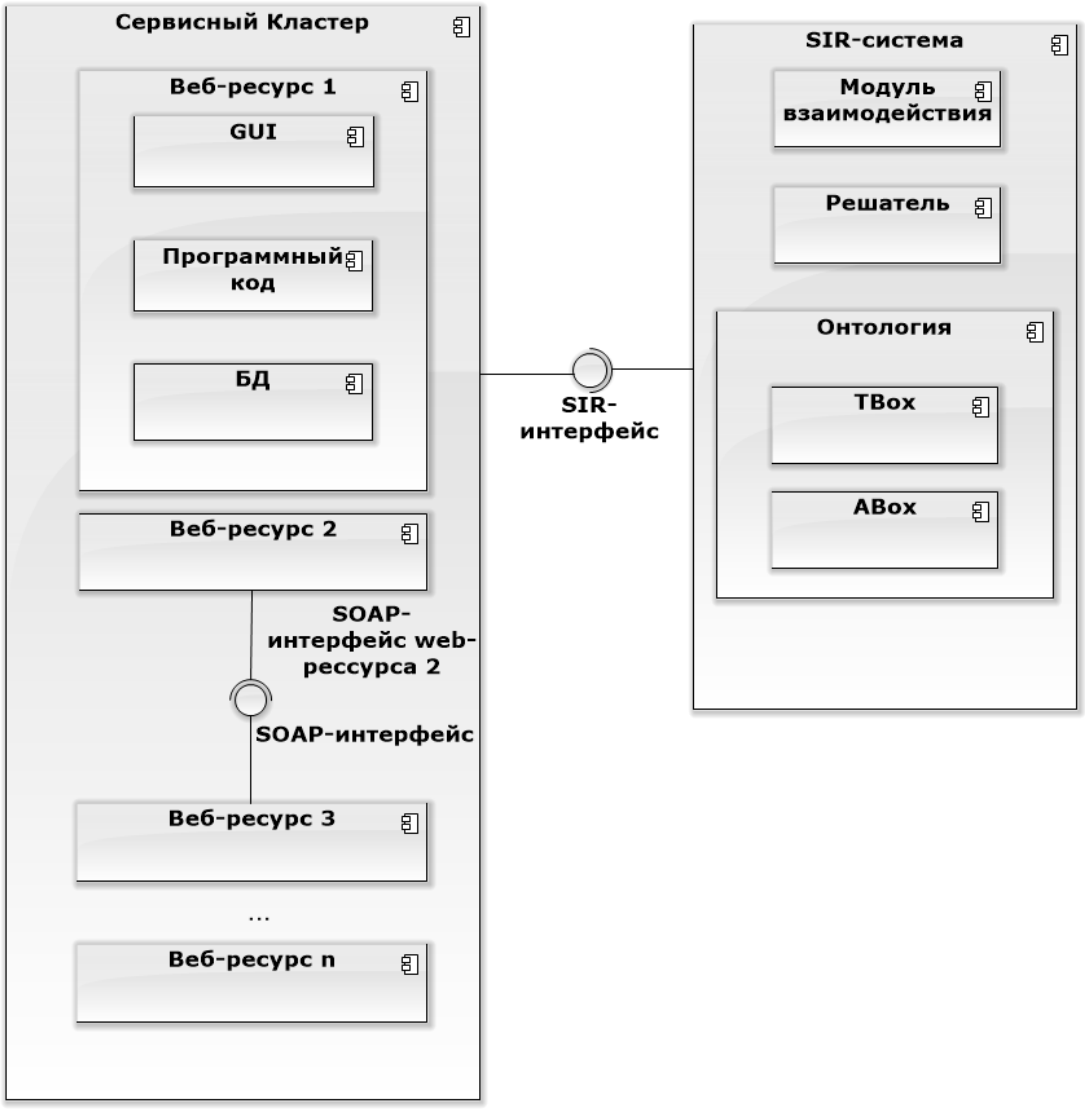
# ГЛАВА 4. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ СИСТЕМЫ АВТОМАТИЗИРОВАННОГО ПОСТРОЕНИЯ ИНТЕРФЕЙСОВ WEB-СЕРВИСОВ

**Средства разработки и хранения данных:**

JAVA SE, PHP, MySQL.

**Минимальные системные требования:**

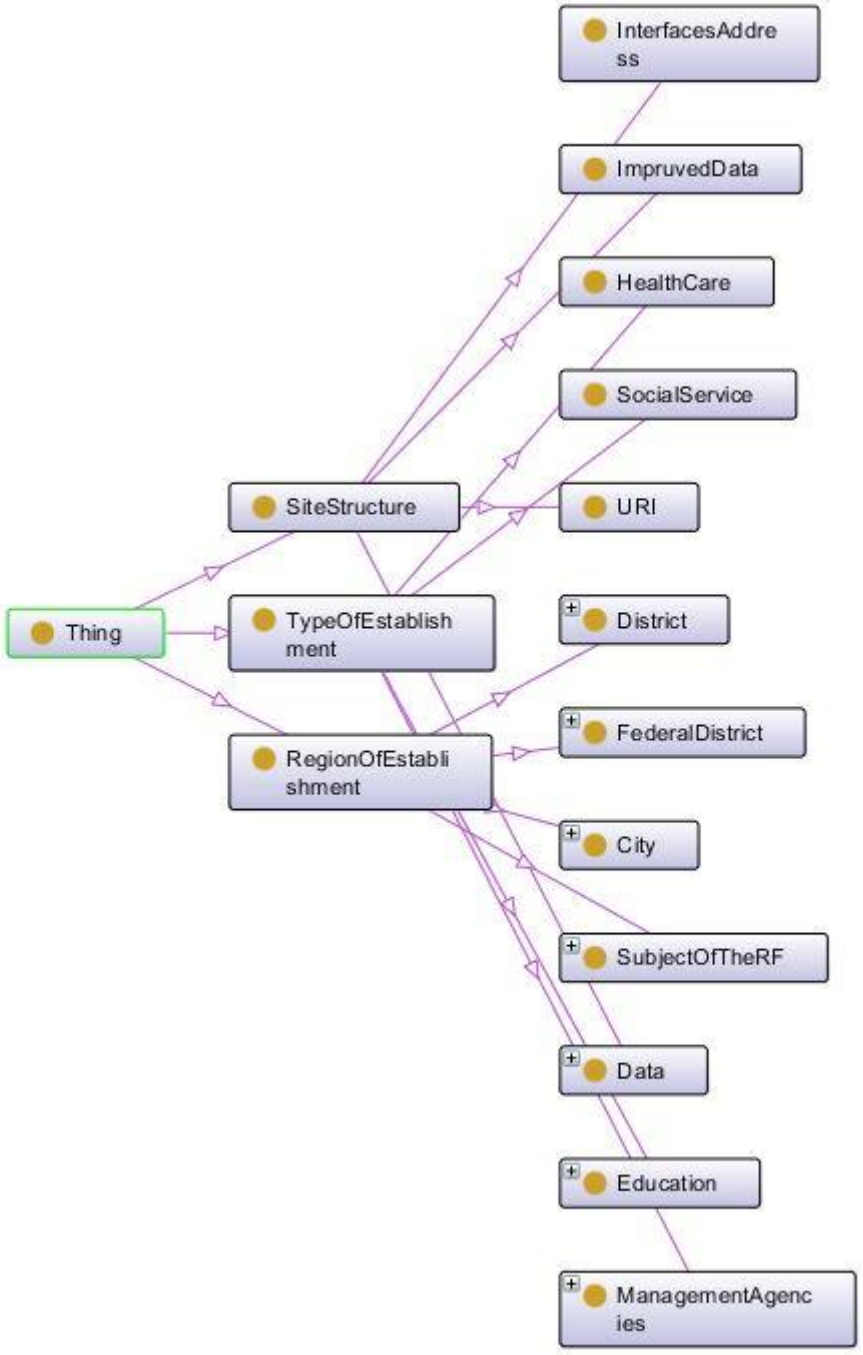
- процессор Intel Pentium 4 или AMD-аналог;
- объем ОЗУ 1 Гб;
- объем дисковой памяти 100 Гб и выше (с ростом данных в БД).
- операционные системы семейства Windows, JRE-совместимые Unix-системы;
- сервер баз данных MySQL 5.1;
- JRE 6.0.



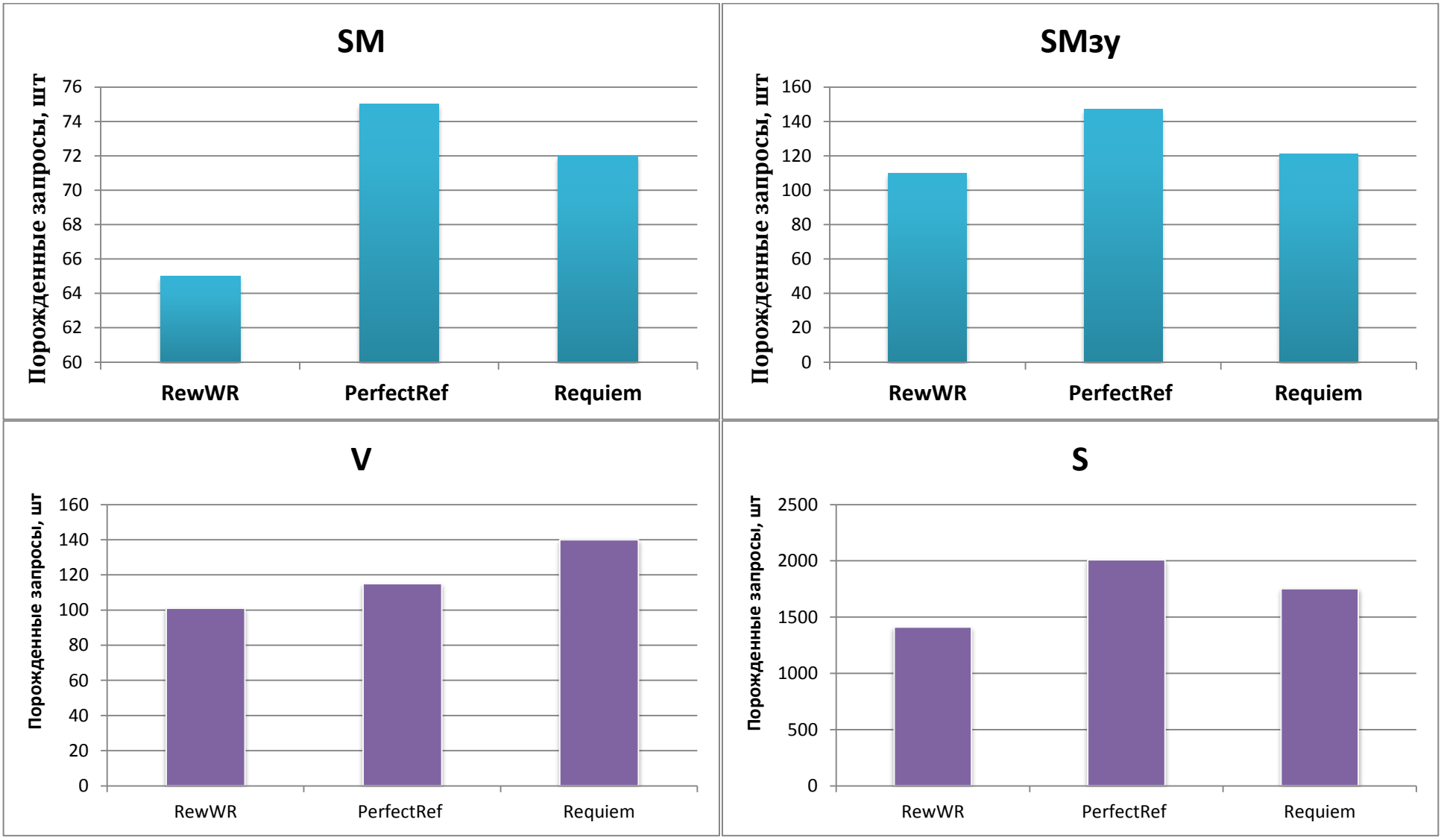
# Пример сгенерированного программного кода web-сервиса

```
<?php
class QuoteService {
   private $anketa = $GLOBanketa;
   function queryStatement($lastname) {
         return $this->anketa["name"];
          if ((isset($this->anketa[$lastname]))&(isset($this- >anketa[$name]))&
         (isset($this->anketa[$middlename]))&(isset($this->anketa[$login]))) {
               return $this->anketa[$name];
         } else {
               throw new SoapFault("Server","Unknown Symbol '.$name'.");
} } }

ini_set("soap.wsdl_cache_enabled", "0");
$server = new SoapServer("DouRegistrationService.wsdl");
$server->setClass("QuoteService");
$server->handle();  ?>
```


InterfacesAddre
ss
ImpruvedData
HealthCare
SocialService
SiteStructure
URI
Thing
TypeOfEstablish
ment
District
FederalDistrict
RegionOfEstabli
shment
City
SubjectOfTheRF
Data
Education
ManagementAgenc
ies

# Сравнение алгоритмов трансформации



# ОСНОВНЫЕ РЕЗУЛЬТАТЫ РАБОТЫ

Зафиксировано среднее уменьшение трудоемкости создания веб-сервисов на базе веб-сайтов примерно на 23 %

**Внедрения**

SIRSystem v1.0

в УОМП г. Рязани

**Публикации**

16 работ: 7 статей(3 по списку ВАК), 9 тезисов докладов

## Научная новизна исследований

- ✓ Полученные алгоритмы автоматизированного построения интерфейсов web-сервисов для SOA-архитектуры отличаются от аналогов скоростью работы и более общей моделью.
- ✓ Получены новые CRUD-алгоритмы для SOA-системы.
- ✓ Получен метод оптимизации алгоритмов трансформации запросов под SOA-систему, отличающийся от более общих алгоритмов трансформации запросов более быстрой работой за счет игнориров ания несущественных элементов входных данных.