Санкт-Петербургский государственный политехнический университет
Факультет технической кибернетики
Кафедра компьютерных систем и программных технологий

Разработка методов и средств автоматизации реинжиниринга устройств, заданных HDL-спецификациями

Выполнил студент гр. 6081/12 О. В. Ненашев
Руководитель, ст. преп. С.Л. Максименко
Научный консультант, к.т.н. А.С. Филиппов

Санкт-Петербург
2011

Введение

Реинжиниринг - это систематическая трансформация существующей системы с целью улучшения ее характеристик. [1]

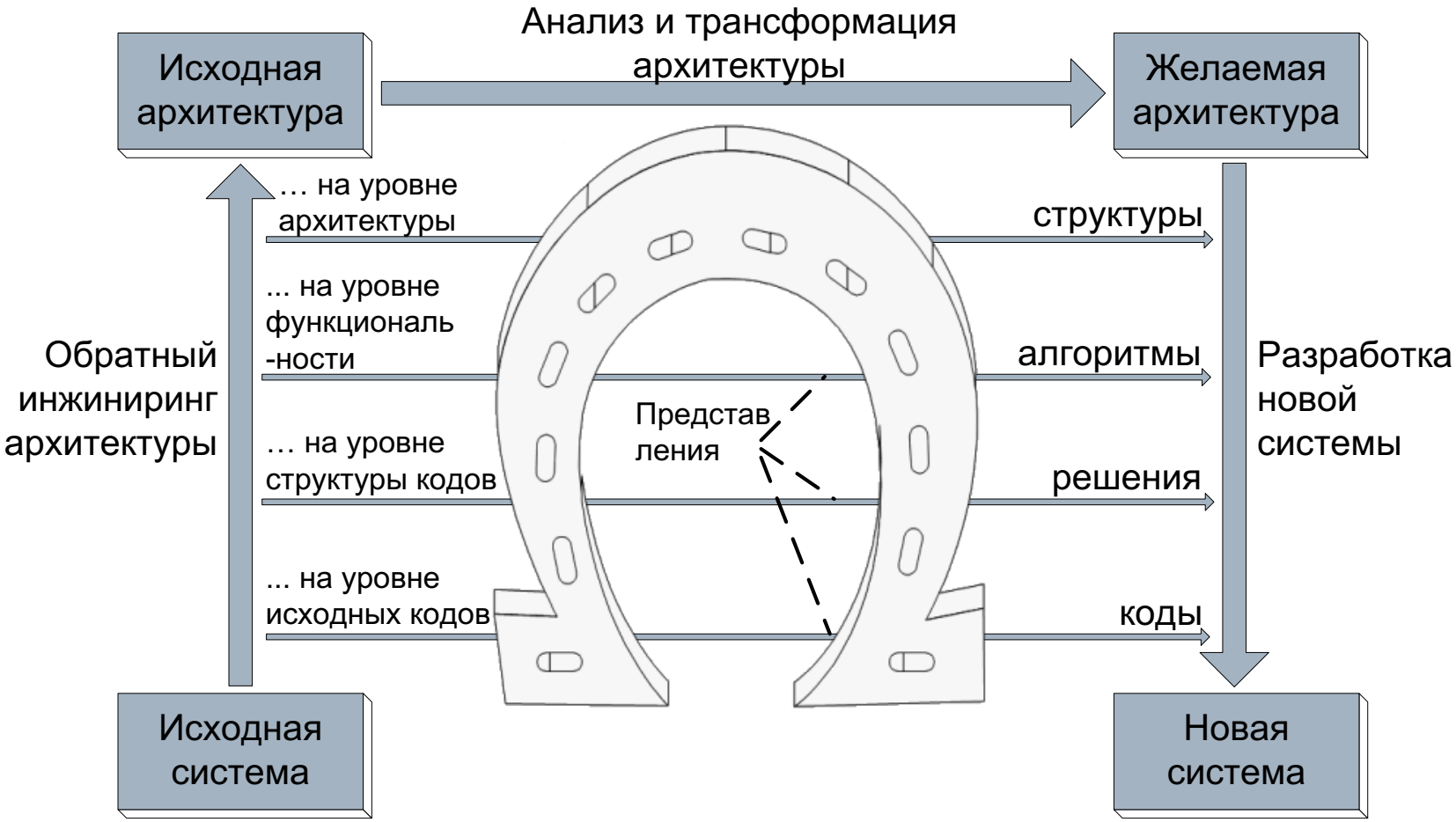
Реинжиниринг...

- ... существовал всегда
- … актуален для сложных устройств
- … требует квалифицированных разработчиков
- … дорог и длителен
- … включает множество рутинных операций

⇨Актуальна задача автоматизации реинжиниринга

⇨Средства автоматизации реинжиниринга (CAP)

Порядок проведения реинжиниринга
Модель “подковы” [2]



Реинжиниринг устройств

- Реинжиниринг устройств актуален
- Популярны специализированные языки описания устройства (HDL*)

Особенности реинжиниринга:

- HDL является спецификацией устройства
- Не требуется восстановление и реализация архитектуры
- Требуется перенос между представлениями
- Сохраняется задача анализа и трансформации архитектуры

*HDL - Hardware Description Language

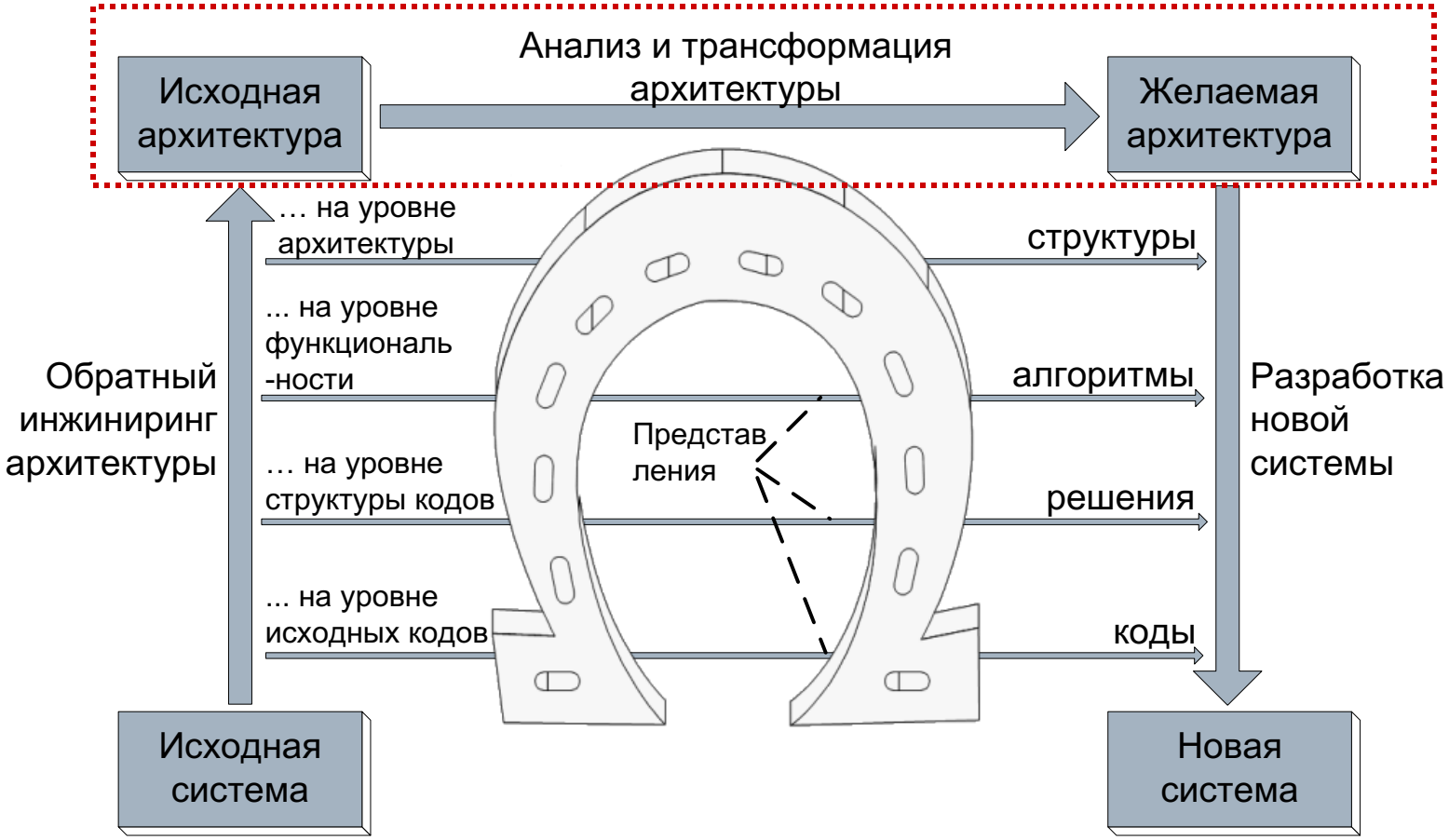
Характеристики устройства:

- функциональность
- производительность
- стоимость
- надёжность
- тестопригодность
- …

Примеры HDL:

- VHDL
- Verilog
- SystemC
- UML
- …

Порядок проведения реинжиниринга
Модель “подковы” [2]



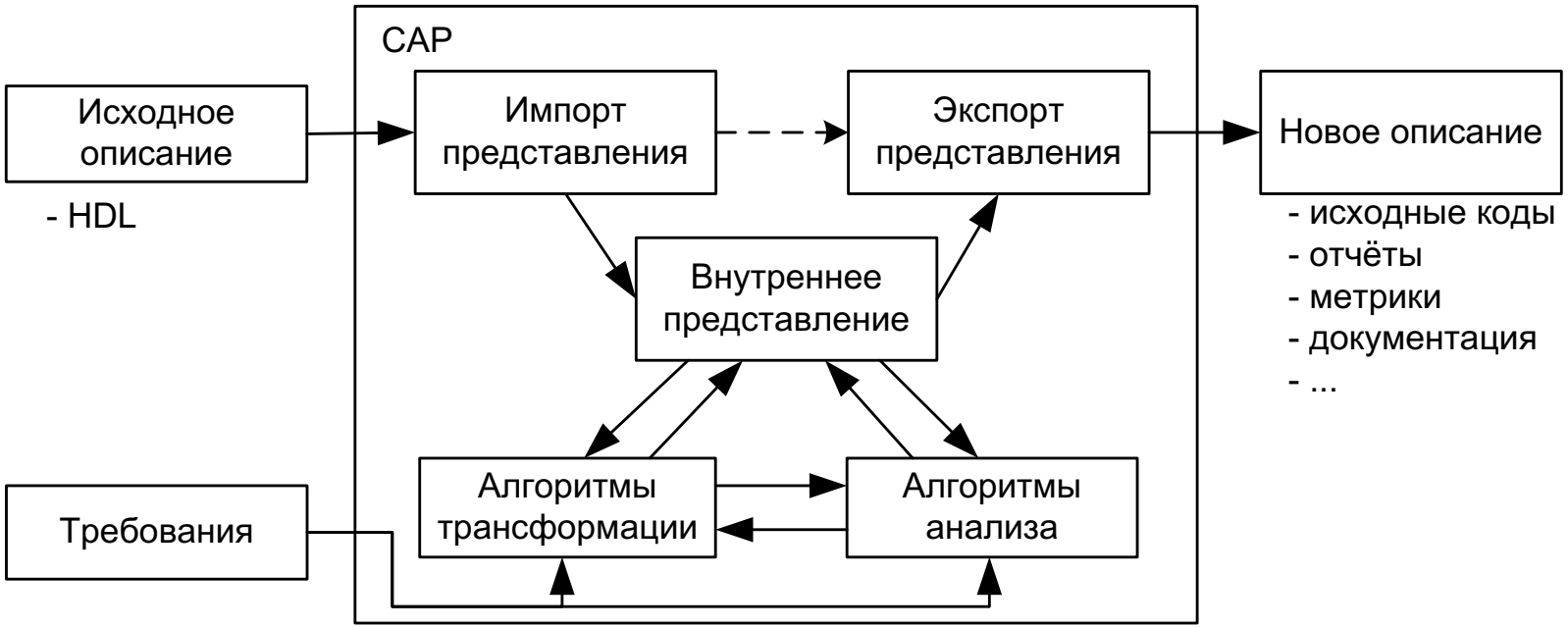
Примеры задач реинжиниринга устройств

- Изменение характеристик устройства
  - Добавление новой функциональности
  - Повышение надёжности
    - Замена блоков отказоустойчивыми аналогами
    - Внесение структурной избыточности и резервирования
    - Блоки самодиагностики устройства
  - Повышение производительности
    - Конвейеризация
    - Использование специальных аппаратных ресурсов
  - Повышение тестопригодности

Примеры задач реинжиниринга устройств

- Перенос устройства между представлениями
  - Перенос устройства с ПЛИС на заказную СБИС
  - Перенос спецификации устройства на другой HDL
  - Конвертация нетлистов в синтезируемые HDL
  - Реверс-инжиниринг архитектуры
- Рефакторинг представления
  - Переименование элементов описания
  - реструктуризация кода
  - изменение иерархии модулей в описании
  - комментирование исходных кодов представления

Пример CAP устройства, описанного на HDL



Задачи:

- Импорт из HDL/экспорт представления
- Анализ архитектуры
- Трансформация архитектуры

Существующие решения по автоматизации реинжиниринга

- Средства рефакторинга HDL
  - Sigasi HDT
  - RAMS (для VHDL-AMS)
- Специализированные средства
  - MODCO (UML -> структурное описание)
  - САПР Alliance (Поведенческое -> структурное описание)
  - SAVANT/TyVIS (Анализатор + Генератор)
- Частично программируемые средства
  - AMIQ DVT
  - DMS Software Reengineering Toolkit (SRT)

ь Отсутствуют универсальные решения

UML (Unified Modeling Language) – Унифицированный язык моделирования

Проблемы построения универсального средства

- Невозможно предусмотреть все пользовательские задачи трансформации архитектуры
- Невозможно обеспечить поддержку всех средств разработки и форматов данных

X Нельзя построить полностью универсальное средство

➢ Возможно предоставить базовый инструментарий, который можно расширить для пользовательских задач

Пытаемся строить САР, которое может быть расширено для решения пользовательских задач реинжиниринга

Основные требования к CAP

- Программируемость
  - Реализация алгоритмов преобразования
  - Реализация алгоритмов анализа
- Расширяемость
  - Расширение внутреннего представления
  - Возможность добавления модулей в CAP (библиотеки преобразований, расширенный анализ, ввод-вывод данных, интерфейсы)
- Встраиваемость
  - Возможность включения средства в процессы разработки
  - Интеграция с другими средствами разработки
- Относительная простота использования

Постановка задач на разработку CAP

Для построения CAP требуется:

- Модель представления устройства
- Механизмы работы с моделью
- Язык управления преобразованиями
- Архитектура средства автоматизации реинжиниринга

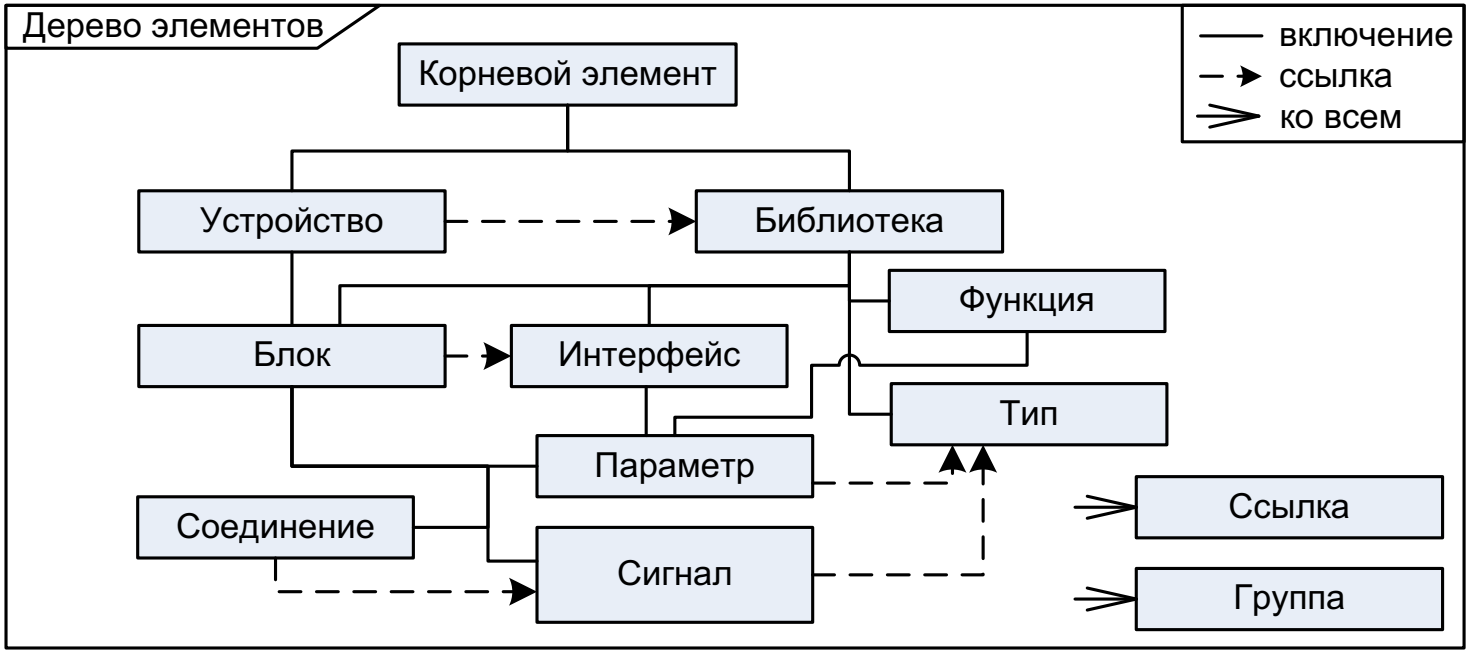
В работе:

- Сформированы частные требования к элементам CAP
- Проведён обзор существующих методик представления и трансформации устройства
- Показана необходимость создания нового представления

- Решено разработать новое представление
- Необходима разработка нового языка преобразования
- Нужно построить архитектуру CAP

Модель представления устройства

- Предназначена для структурного описания устройства
- Методика основана на архитектурном графе
- Используется ограниченное число базовых ячеек



Свойства представления устройства

Особенности

- Древовидная иерархия элементов
- Использование ссылок для связей между уровнями
- Механизмы группировки элементов
- Параметризация ячеек
- Абсолютная и косвенная адресация к элементам дерева
- Возможность верификации дерева элементов

Расширяемость

- Добавление параметров
- Наследование от базовых типов ячеек

Язык управления преобразованиями

Требования
- Полнота преобразований устройства
- Получение всей информации о представлении
- Управление средством реинжиниринга

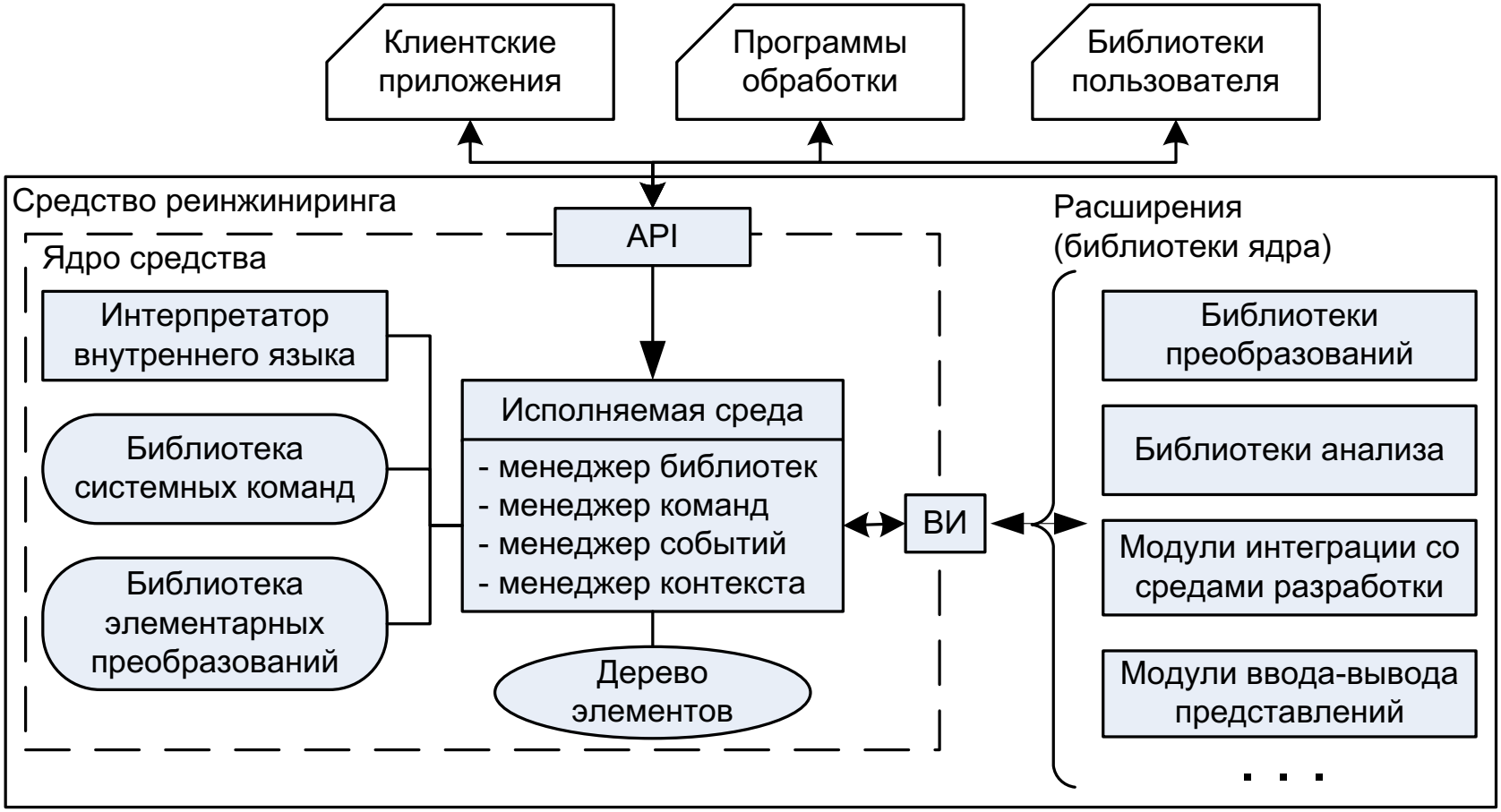
Базовые команды работы с деревом элементов:
- Добавление/удаление/копирование ячеек
- Установка и считывание значений параметров
- Получение списков ячеек и параметров для элементов
- Навигация по дереву элементов
- Верификация ячеек

Команды управления CAP:
- Управление библиотеками пользователя
- Запуск программ обработки
- Работа с историей команд

Разработка архитектуры CAP

- Модульная архитектура
- Минимальное ядро с базовой функциональностью
  - Дерево элементов
  - Системные библиотеки команд
  - Модули управления контекстом
- Прочая функциональность реализуется через расширения
  - Пользовательские команды обработки
  - Модули ввода-вывода данных
  - Интеграция со сторонними средствами
  - Поддержка внешних языков программирования
  - Дополнительное API
- Взаимодействие через API
  - Вызов команд и получение результатов
  - Механизм событий

Структура модульного CAP



Разработка прототипа САР

- Прототип ориентирован на Quartus II и ПЛИС семейства Cyclone 2.
- Встраиваемся в процесс разработки в среде Quartus
- Входной формат – VHDL-нетлисты, выходной – VHDL
- Функциональность ядра реализуется частично

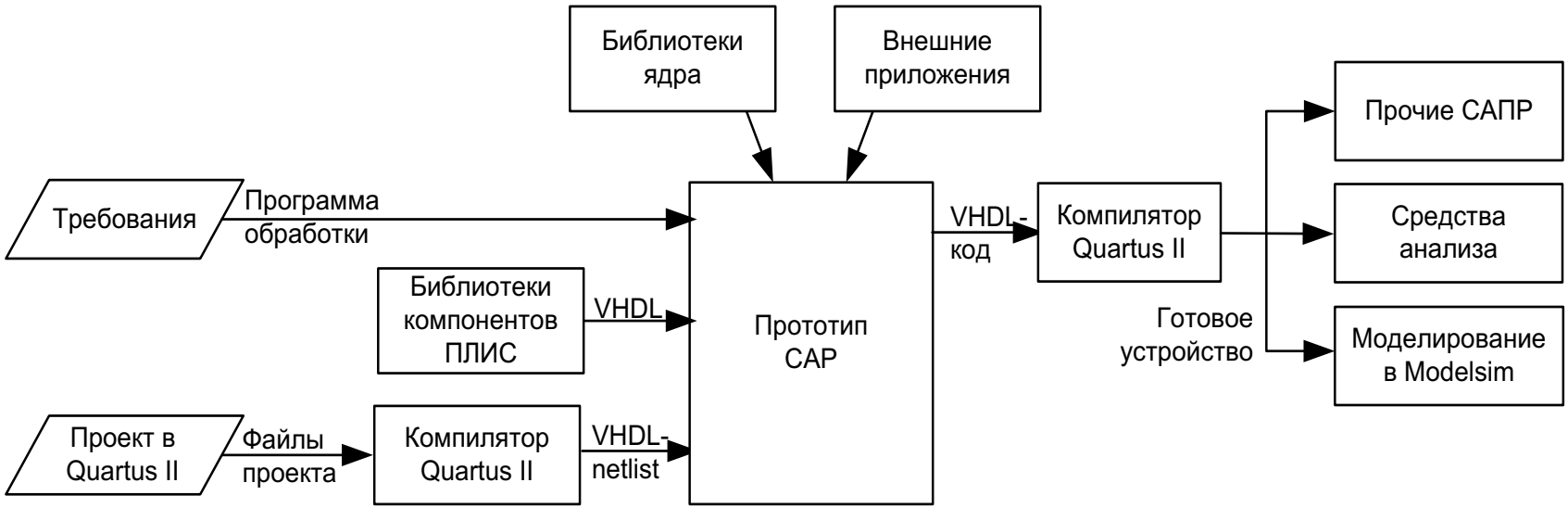


Рис. Схема взаимодействия прототипа САР со средой Quartus 2

Состав прототипа

Средство реинжиниринга:

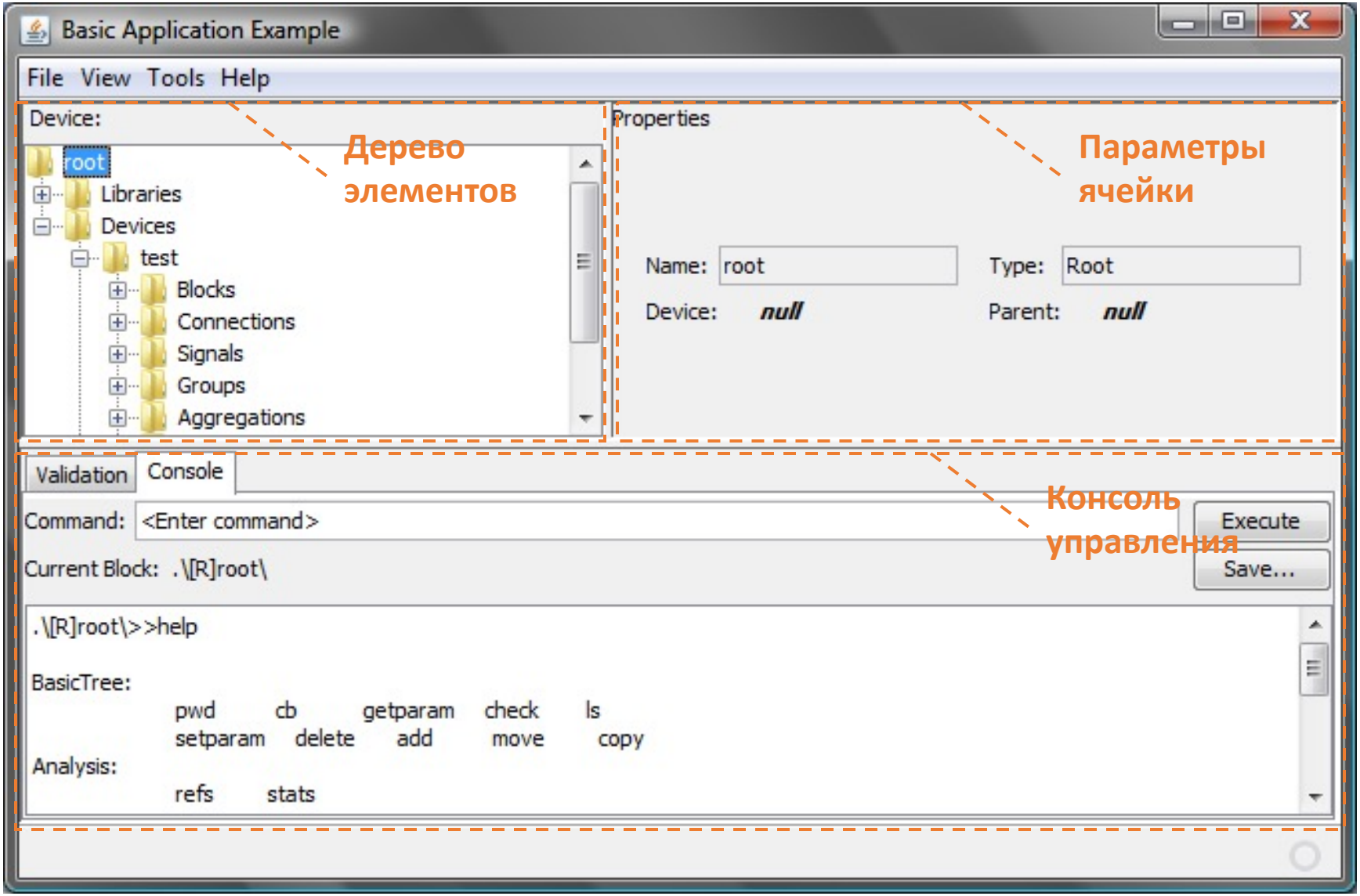
- Прототип ядра CAP
- Библиотека ввода-вывода VHDL-нетлистов
- Библиотека работы с VHDL

Дополнения:

- Консольный интерфейс пользователя
- Графический интерфейс пользователя
- Модуль с шаблонами для NetBeans
- Библиотека введения структурной избыточности в устройство
- Библиотека сбора статистики по дереву элементов

- Для организации совместного взаимодействия развёрнута система управления проектами на базе Redmine
- Общий размер исходных кодов - ~50,000 строк

Графический интерфейс прототипа



Апробация прототипа

- Проведён эксперимент с введением структурной избыточности в устройства => программируемость
- Подтверждена возможность взаимодействия со средой Quartus II => интегрируемость
- Средство реинжиниринга встроено в тестовые пользовательский и графический интерфейсы => встраиваемость
- Егоровым Е.В. разработана библиотека сбора статистики по дереву элементов => расширяемость

- Выполнены основные требования к САР
- Подтверждена применимость подходов
- Выявлены недостатки и предложены пути их решения

Направления дальнейшей разработки

Доработка методик

- Поддержка поведенческих описаний в представлениях
- Расширенное управление типами
- Полноценное представление функций
- Упрощение механизмов соединений

Доработка архитектуры САР

- Управление правами доступа к элементам ядра
- Введение механизмов защиты от ошибок
- Механизмы работы с проектами

Прототип САР

- Доработка механизмов ядра до полного соответствия архитектуре и требованиям
- Разработка специализированных библиотек для решения частных задач реинжиниринга
- Расширение интеграции средства со средой разработки
- Разработка API и поддержка внешних языков программирования
- Доработка до полноценной САПР

Выводы

- Исследованы подходы к построению САР и существующие решения по представлению устройства
- Показана актуальность разработки программируемого САР
- Сформированы требования к средству
- Разработана методика представления устройства и язык управления преобразованиями
- Построена архитектура САР (программируемая, расширяемая, встраиваемая)
- Разработан прототип САР
- Проведена апробация прототипа и подтверждена применимость подходов
- Предложены пути для дальнейшей разработки

Направления дальнейших исследований

- Совместный реинжиниринг структурных и поведенческих описаний
- Реинжиниринг цифро-аналоговых устройств
- Реинжиниринг при сохранении связей с исходным представлением

ь Разработка и исследования будут продолжены в аспирантуре

КОНЕЦ

Спасибо за внимание!

Дополнительная информация:

- https://nenhome-apps.sourcerepo.com/redmine/nenhome/
- Пояснительная записка к диссертации
- Автор работы

Ответы на вопросы рецензента

Замечания

Замечание 4

- Основная задача раздела 2 – поиск направлений для разработки => предварительное исследование

Замечание 7

- Обзор средств проведён для выявления непокрытой ими области
- Только DMS проходит по критерию программируемости

Замечание 8

- Имело смысл использовать EDIF, но решающим фактором стала возможность расширения до поддержки VHDL во входных форматах
- Выходной формат прототипа близок к нетлистам, но совместим с VHDL
- Для VHDL взят наиболее простой способ встроиться в процесс разработки

С прочими замечаниями согласен, часть из них устранена в новой версии диссертации

1. Внутренний язык средства

> Неясно, зачем потребовалась разработка собственного “внутреннего языка управления средством”. Существует ряд скриптовых языков, которыми можно было воспользоваться. Рецензент не нашёл описания внутреннего языка

- Планировалась реализация полноценного языка программирования на базе одного из скриптовых языков (основной кандидат - Tcl)
- При разработке было решено оставить реализацию языков программирования CAP для расширений
- Внутренний язык реализует взаимодействия ядра средства со своими расширениями
- В качестве внутреннего языка используется Java с доступом к ядру средства через API через систему команд, описанную в пункте 3.3 и приложении А

2. Архитектурный граф (АГ) и Абстрактный семантический граф (АСГ)

> В чём отличие модели архитектурного графа от АСГ? Можно ли дополнить модель АСГ так, чтобы она превратилась в модель АГ? Если да, то почему в п. 2.3.3. использование модели АСГ выделяется как недостаток?

- Любые знания можно представить в виде совокупности объектов (понятий) и связей (отношений) между ними. [3] => в виде семантического графа
- Архитектурный граф можно представить в виде семантического
- Отличия АГ.
  - Не является абстрактным
  - В узлах АГ лежат комплексные объекты
  - Узлы могут включать методы обработки и т.п.
- АГ удобнее для анализа и преобразования => использование АСГ является недостатком

3. Отличие системных команд от базовых операций

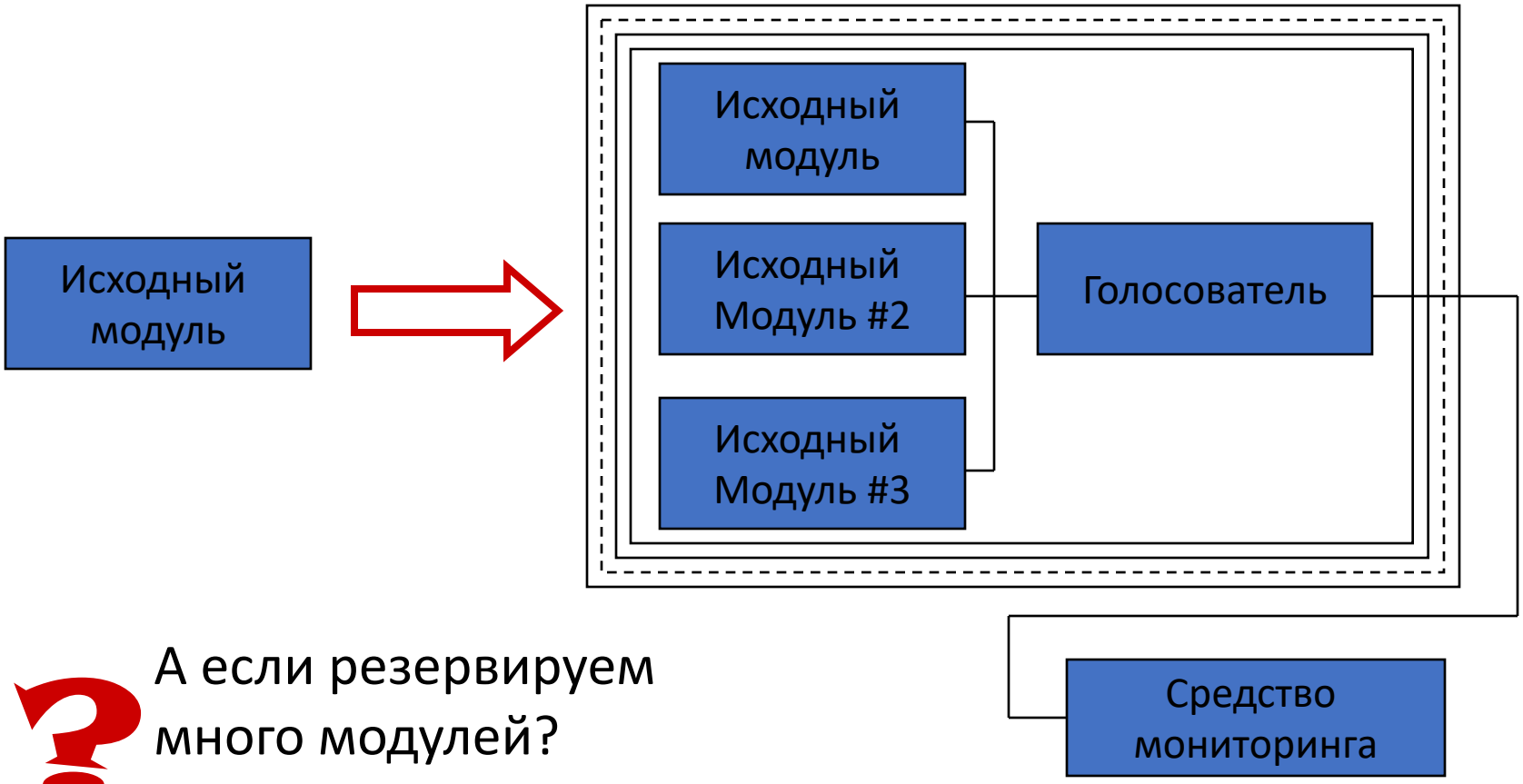
Поясните, пожалуйста, отличие системных команд от базовых операций (подраздел 3.4.7, библиотеки SystemLib и BasicLib)

Системные команды (биб-ка SystemLib)	Базовые операции (биб-ка BasicLib)

• Хранит команды управления ядром средства
  • Управление расширениями
  • Управление историей
  • Запуск программ
  • Выход из средства
  • …
	• Содержит команды работы с деревом элементов
	  • Получение информации об элементах описания
	  • Элементарные операции с деревом элементов
	  • …

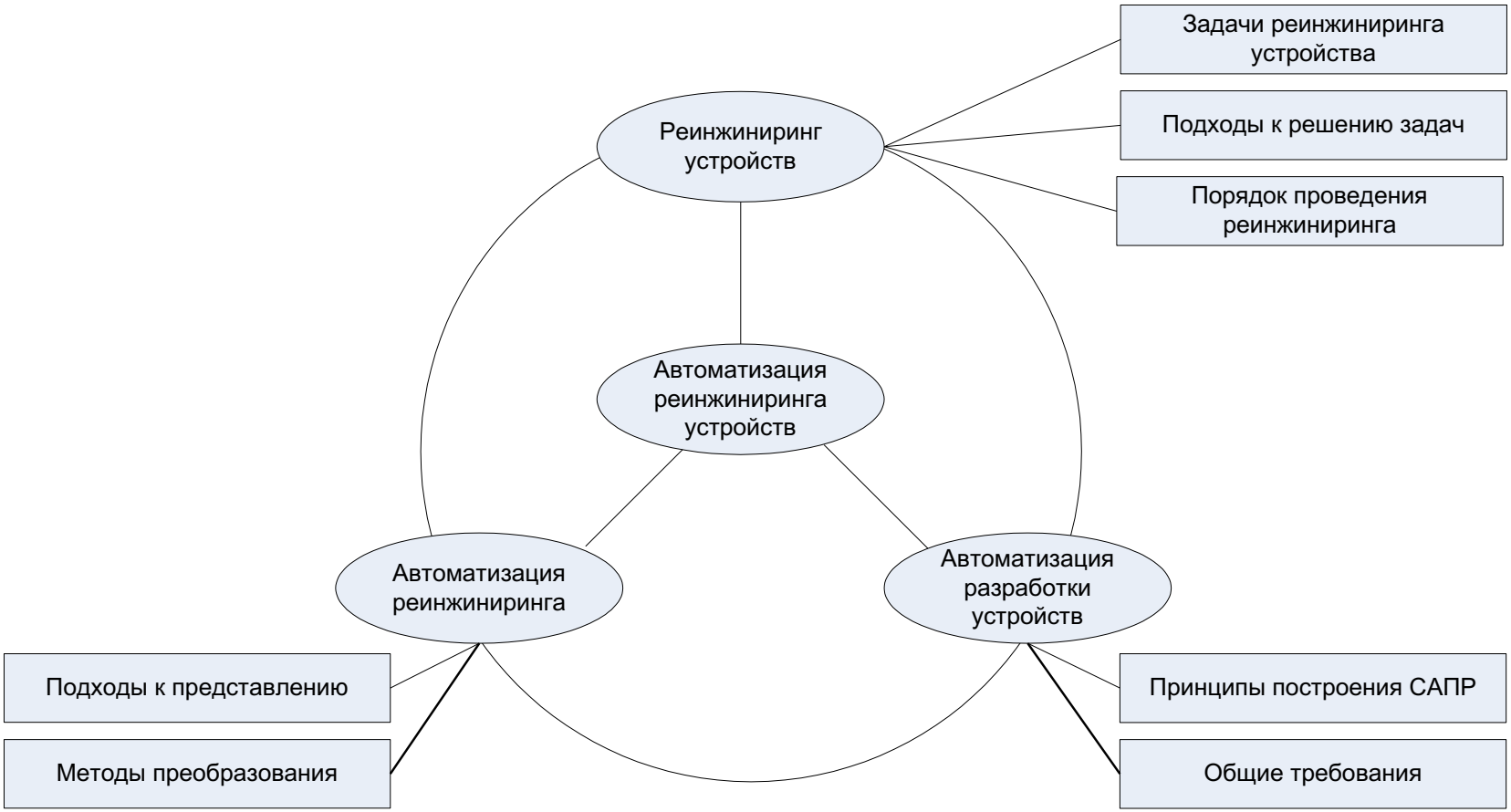
Приложения

Пример. Внесение структурной избыточности в модуль устройства

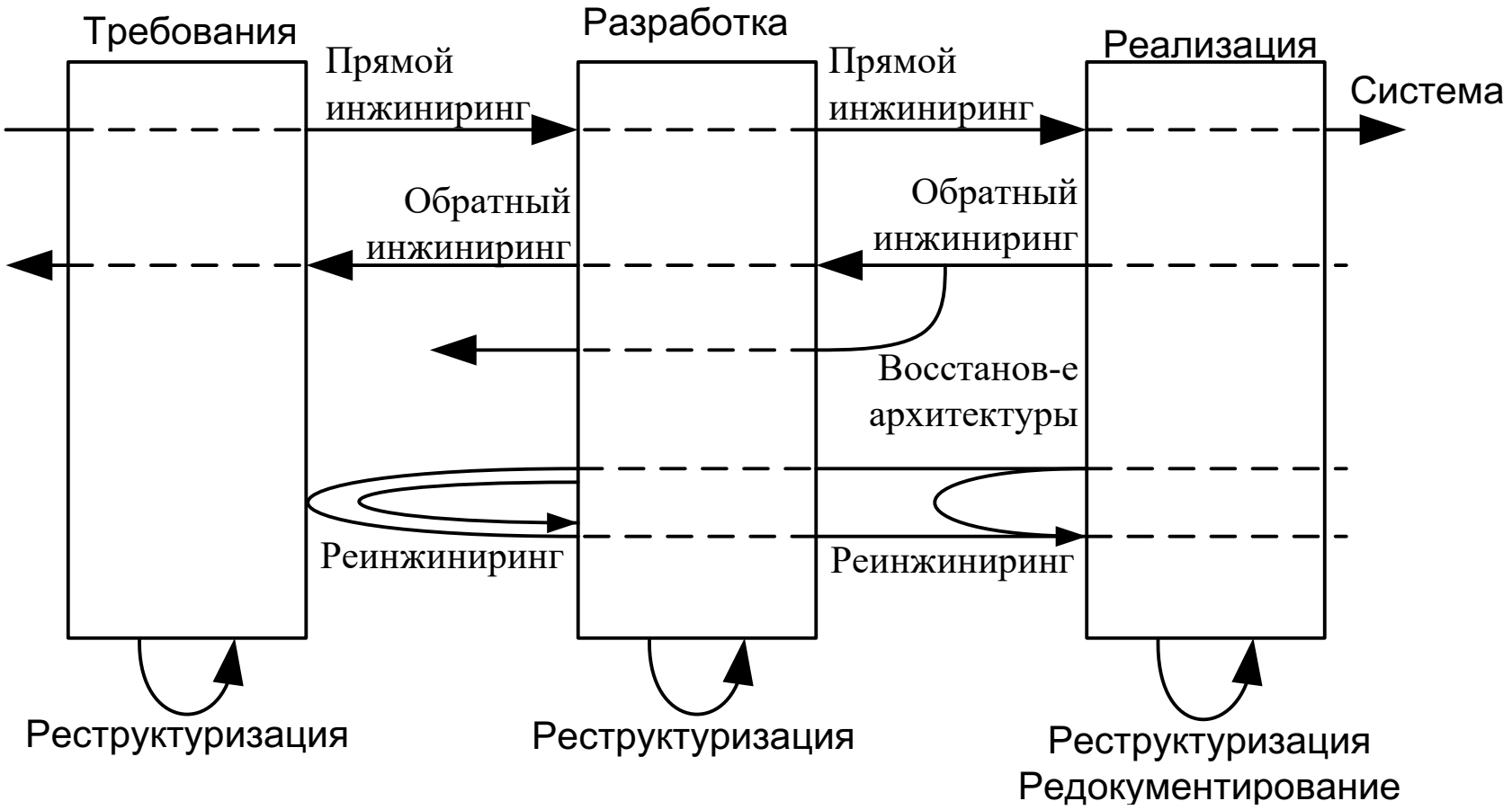


А если резервируем много модулей?

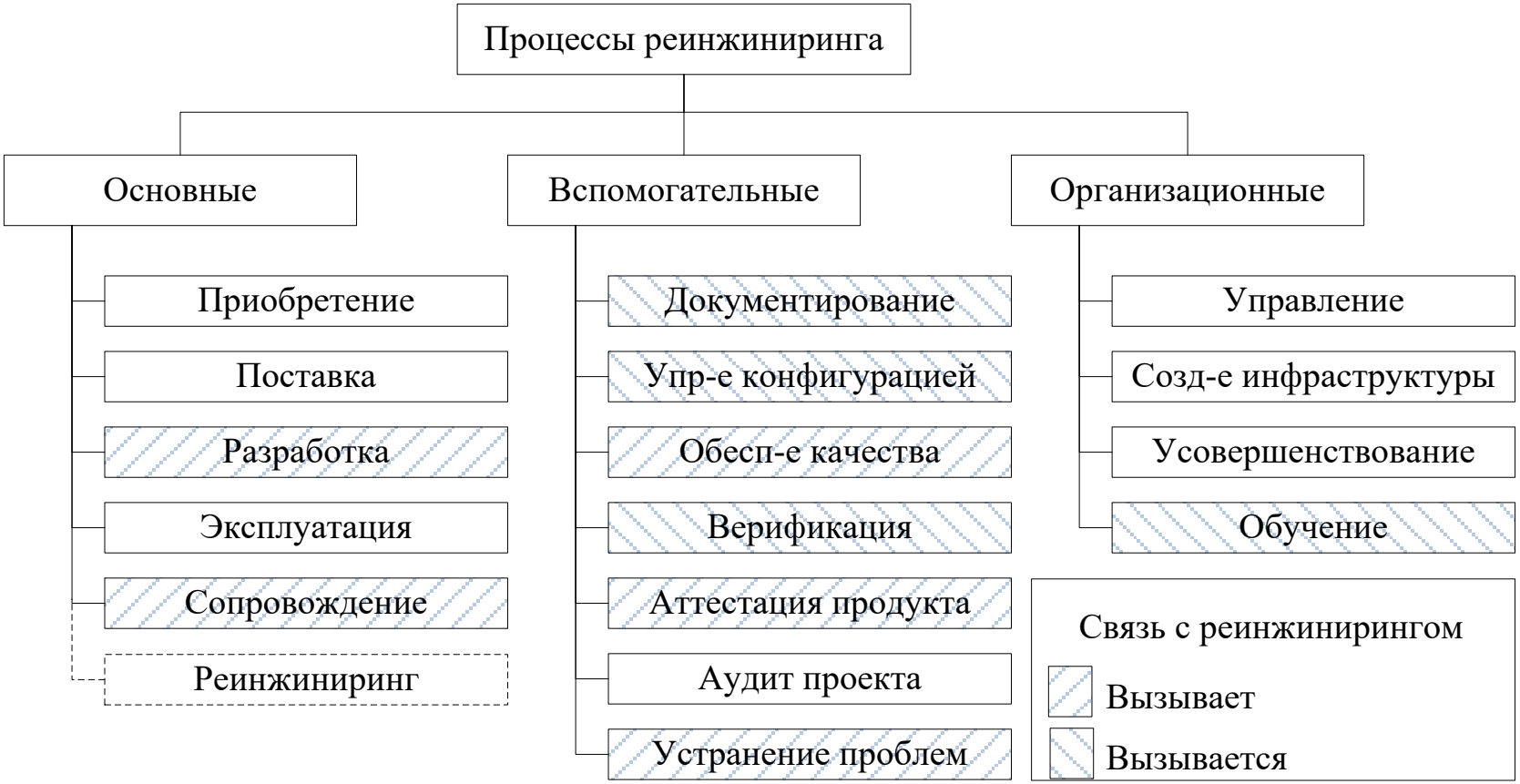
Смежные области



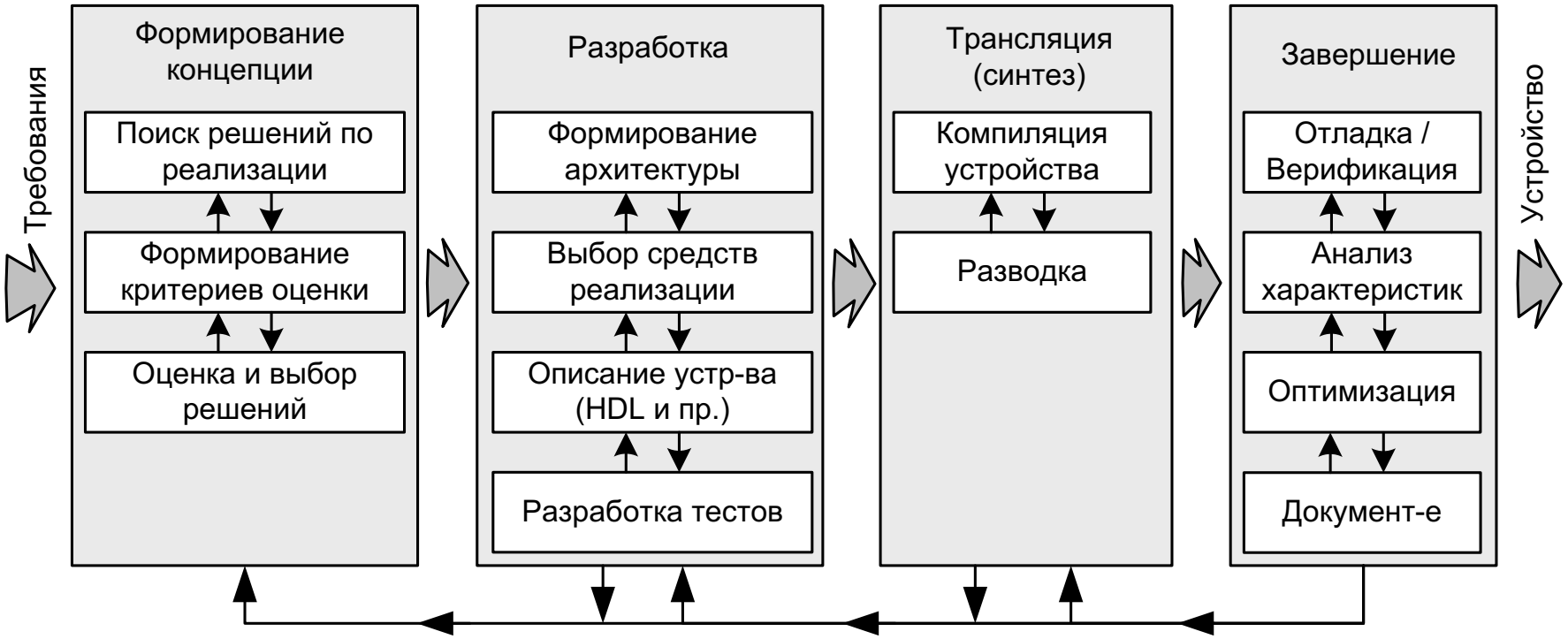
Место реинжиниринга в в процессе разработки системы [4]



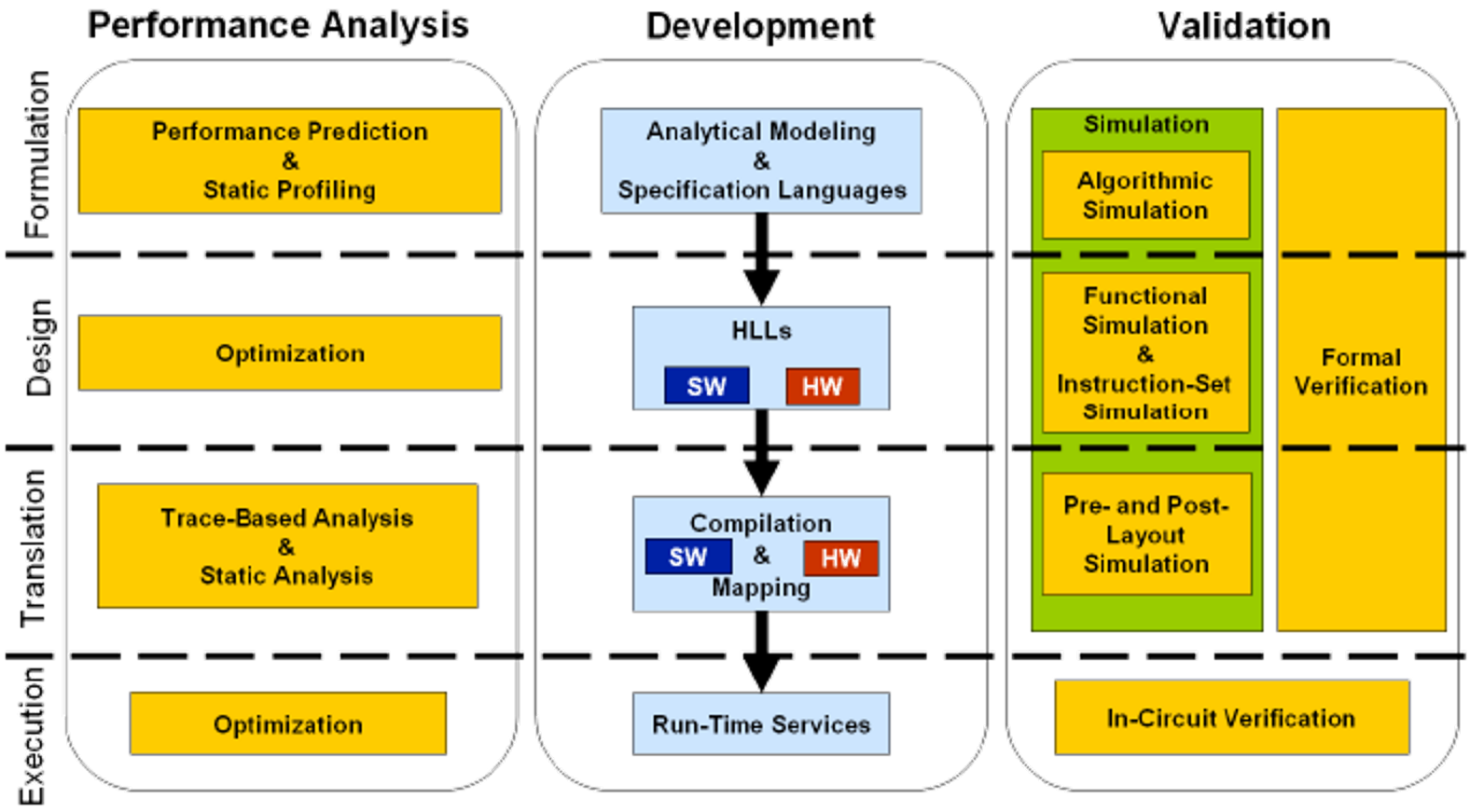
Связи реинжиниринга с процессами жизненного цикла по ISO [5]



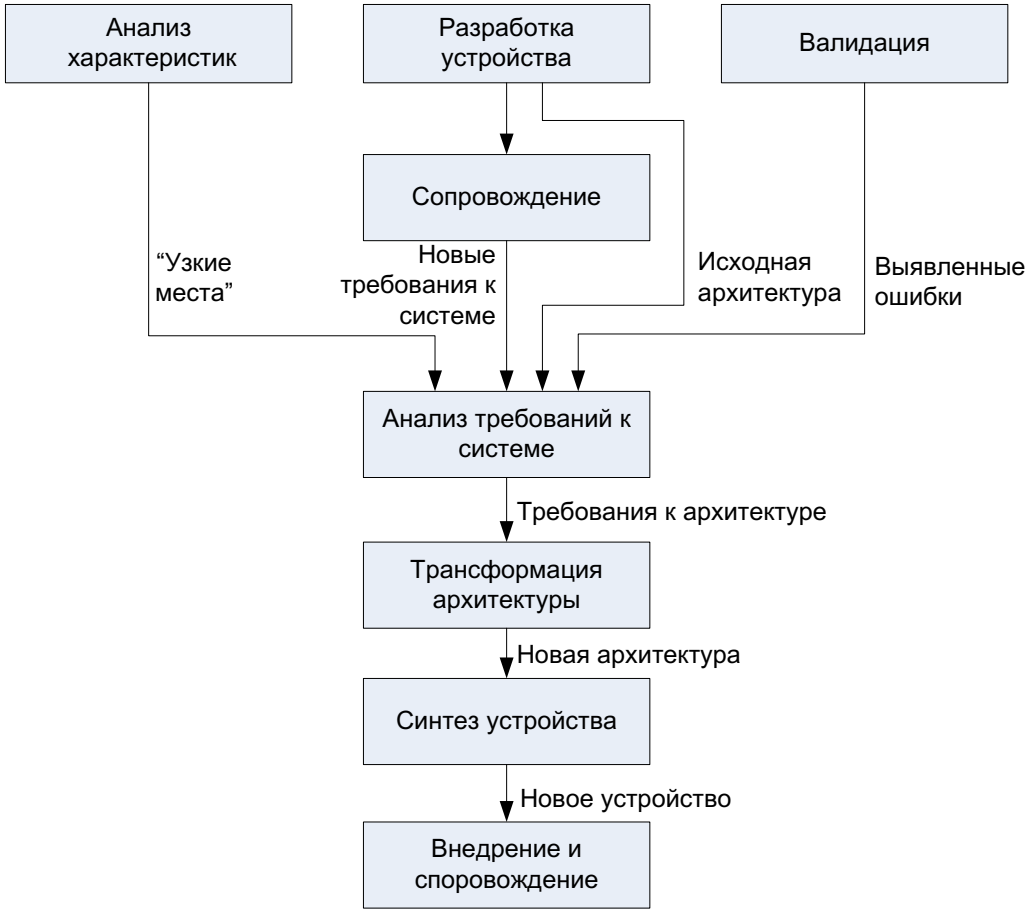
Основные этапы разработки устройства



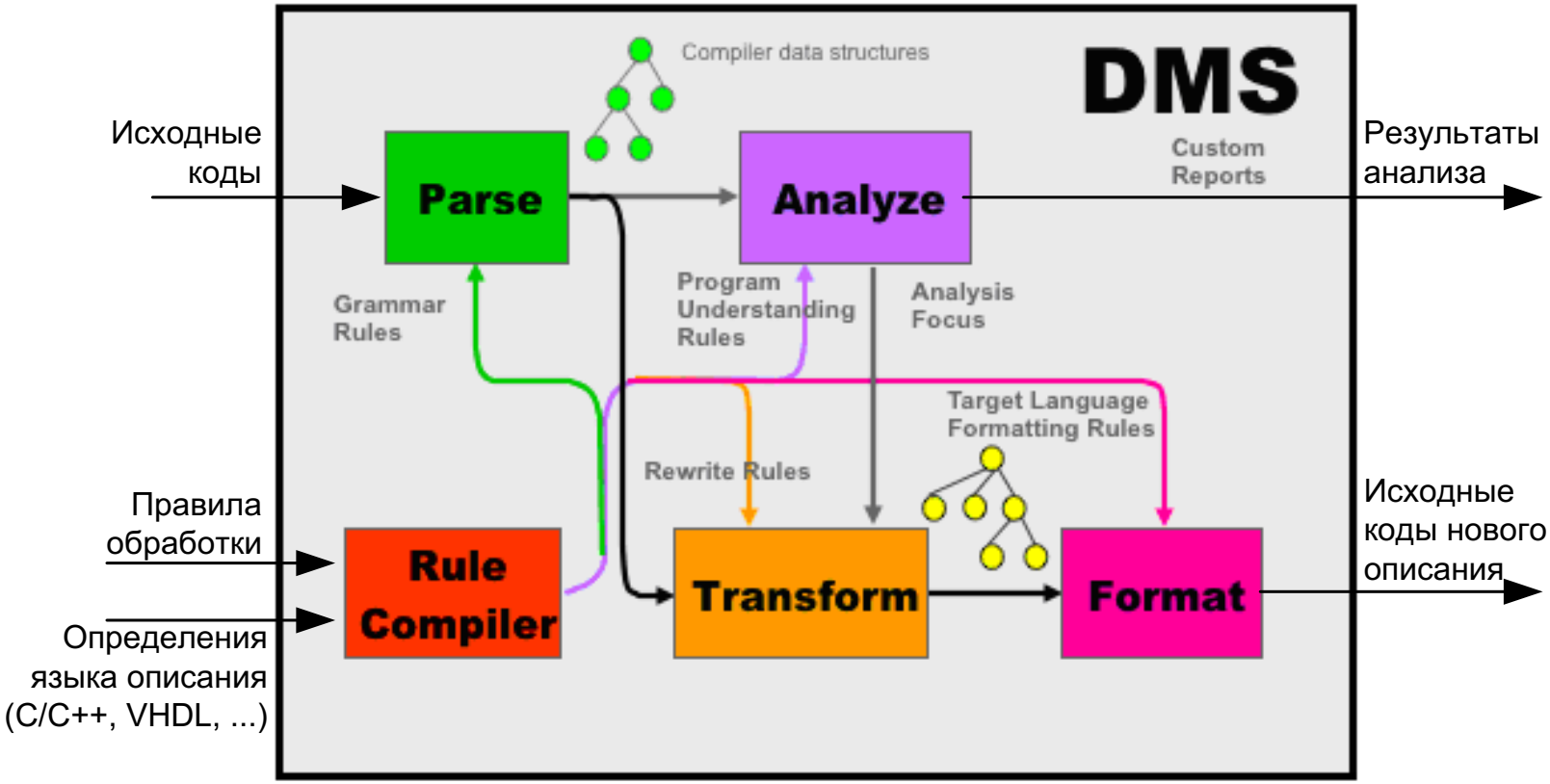
Процессы контроля качества разработки [6]



Включение реинжиниринга в процесс разработки



DMS Software Reengineering Toolkit [7]



DMS Software Reengineering Toolkit (продолжение)

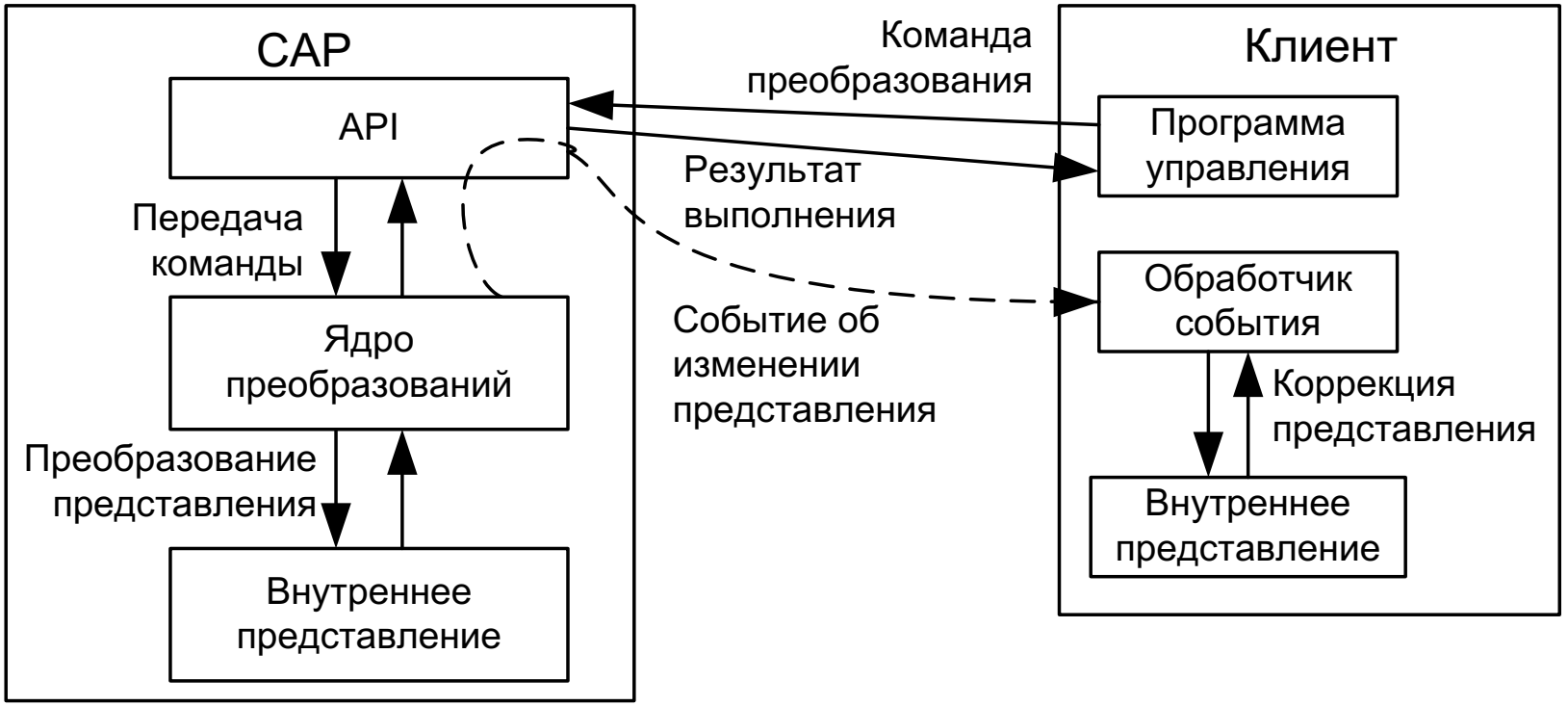
Свойства

- Поддержка различных входных форматов
- Алгоритмы обработки преобразуются во внутренние наборы правил

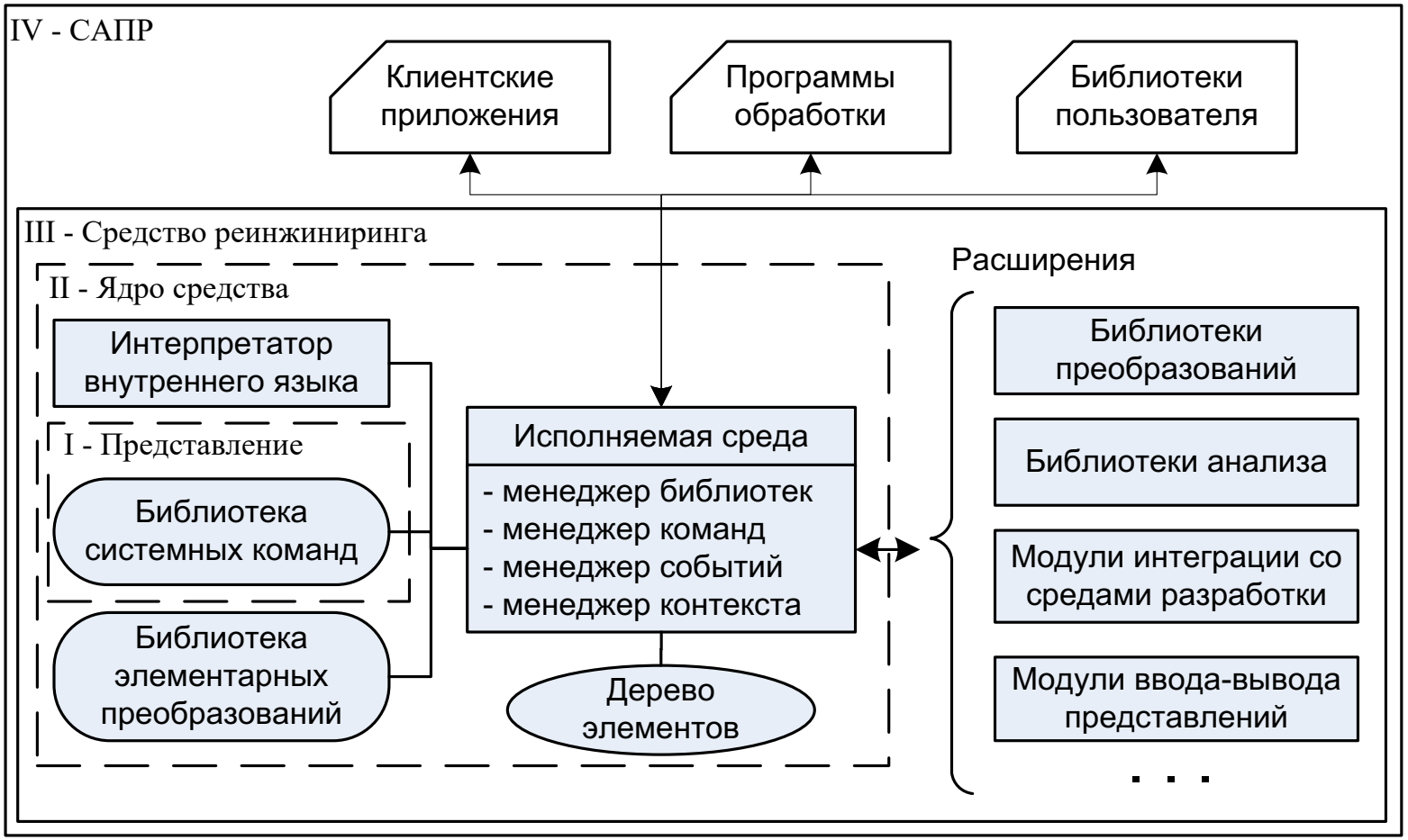
Недостатки при реинжиниринге устройства

- Сложность программ
- Нет возможности взаимодействия с внешними средствами
- Используется для реинжиниринга исходных кодов, но не самого устройства
- Высокая стоимость и закрытость

Пример взаимодействия стороннего приложения с CAP

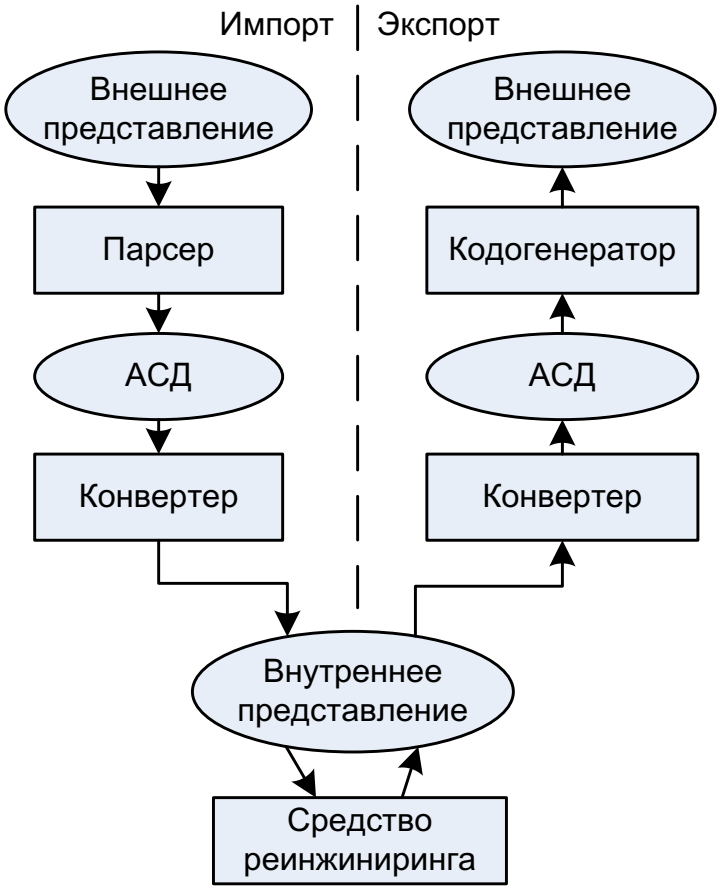


Уровни использования прототипа в сторонних средствах



Используемый подход к преобразованию

- Используем промежуточное представление в АСД
- Разбиваем преобразование на две части
- Каждую часть можно заменить



Опробованные задачи реинжиниринга

- Трансформация архитектуры
  - Введение структурной избыточности
  - Удаление несинтезируемых сигналов
  - Восстановление интерфейсов по сигналам
- Перенос представлений
  - VHDL-нетлисты -> ВП*,
  - ВП -> VHDL
  - ВП -> текст
- Рефакторинг представления
  - Переименование элементов
- Анализ
  - Сравнение интерфейсов
  - Получение трасс сигналов

ВП – внутреннее представление

Средства разработки прототипа

Средство	Решаемые задачи
Средства разработки	
NetBeans IDE 6.9	- программная реализация прототипа; - тестирование и отладка прототипа; - пользовательский модуль с шаблонами для библиотек и классов;
Quartus II 10	- разработка тестовых примеров для CAP; - разработка модулей для пользовательской библиотеки повышения надёжности; - генерация нетлистов для прототипа.
ModelSim	- моделирование устройств; - проверка корректности работы устройства, сгенерированного CAP устройства
Enterprise Architect 8	- построения частичной спецификации на средство;

Средства разработки прототипа (продолжение)

Средство	Решаемые задачи
Организация взаимодействия	
Mercurial	- версионирование исходных кодов;
Redmine	- отслеживание ошибок в программной реализации; - управление задачами; - хранилище файлов; - wiki проекта.
Средства документирования	
MS Office 2003	- подготовка программной документации;
NetBeans Javadoc	- генерация документации на исходные коды в стандарте Javadoc.
Doxygen	- генерация документации на исходные коды; - построение диаграмм связей по исходным кодам.
Zotero	- хранение и систематизация библиографии.

Используемые языки разработки

- Java
  - Средство реинжиниринга VHDL
  - Тесты для ПО
- VHDL
  - Специальные блоки для библиотек
  - Тестовые устройства
  - Тесты для устройств
- UML
  - Спецификация на программное обеспечение
- Скрипты
  - Макросы и функции для библиотек
  - Интерфейс к средствам разработки на VHDL

Используемые технологии и библиотеки Java

- JavaCC
  - Импорт нетлистов и VHDL-файлов
  - Импорт специализированных библиотек Quartus
- vhdl_ast
  - Формирование AST для нетлистов Quartus
  - Генерация VHDL из AST
- Swing
  - Графический интерфейс средства
- JUnitTest
  - Тестирование основных модулей средства
- JSAP
  - Разбор командной строки в программах
  - Интерпретатор скриптового языка

Статистика по исходным кодам

Модуль	Число файлов	Размер исходных кодов, строк			
		Код	Ком.	Разд.	Всего
OBL core	127	11514	5817	1650	18981
Библиотеки	37	13611	4661	1743	20015
-- SystemLib	23	1319	313	252	1884
-- BasicLib	15	1517	495	246	2258
-- StatisticsLib	14	1014	237	130	1381
-- ReliabilityLib	12	1571	364	108	2043
-- Библиотеки ввода-вывода	28	8190	3252	1007	12449
Тесты	25	820	76	340	1236
Пользовательские интерфейсы	59	6425	1097	1443	8965
Netbeans module	4	170	162	53	385
Библиотеки элементов (VHDL)	27	2378	324	45	369
Всего	248	32540	12137	5274	49951

Список использованных источников

1. Шестаков А.В. Реинжиниринг // Энциклопедический словарь экономики и права. 2000. с. 568.
2. Ахтырченко К.В., Сорокваша Т.П. Методы и технологии реинжиниринга ИС // Институт Системного Программирования РАН. 2003. 15 с.
3. Ясницкий Л.Н. Теоретические основы информатики // Информатика. 2009. № 17. 1-10 с.
4. Chikofsky E.J., Cross J.H. Reverse engineering and design recovery: A taxonomy // IEEE software. 1990. 13-17 с.
5. Singh R. International Standard ISO/IEC 12207 software life cycle processes // Software Process Improvement and Practice. 1996. т. 2. 35-50 с.
6. Gonzalez I. и др. Classification of Application Development for FPGA-Based Systems // Aerospace and Electronics Conference, 2008. NAECON 2008. IEEE National. с. 203–208.
7. Semantics Designs inc. DMS Software Reengineering Toolkit [Электронный ресурс]. URL: http://www.semdesigns.com/Products/DMS/DMSToolkit.html (дата обращения: 04.11.2010).