МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

На правах рукописи

Хородов Виталий Сергеевич

РАЗРАБОТКА МЕТОДОВ И СРЕДСТВ МНОГОАГЕНТНОГО РАСПРЕДЕЛЕННОГО АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ СТРУКТУРНО-ФУНКЦИОНАЛЬНЫХ ЛИНГВИСТИЧЕСКИХ МОДЕЛЕЙ ВЫЧИСЛИТЕЛЬНЫХ УСТРОЙСТВ

Специальность: 05.13.12 – Системы автоматизации проектирования (промышленность)

Диссертация на соискание ученой степени
кандидата технических наук

Научный руководитель – доктор технических наук, доцент,
Афанасьев Александр Николаевич

Ульяновск – 2015

Содержание

Принятые сокращения и обозначения

ACL – Agent Communication Language

API – Application Programming Interface

CPN – Coloured Petri Nets

CORBA – Common Object Request Broker Architecture

HDL –Hardware Description Language

ISO – International Organization for Standardization

MVC – Model View Controller

PMI – Project Management Institute

VHDL – Very high speed integrated circuits Hardware Description Language

SaaS – Software as a Service

UML – Unified Modeling Language

БД – база данных

ВУ – вычислительные устройства

ППО – промежуточное программное обеспечение

ПССЗ – параллельная сетевая схема задач

САПР – система автоматизированного проектирования

СБИС – сверхбольшая интегральная схема

СВТ – средства вычислительной техники

СРП – система распределенного проектирования

СФЛМ – структурно-функциональная лингвистическая модель

ВВЕДЕНИЕ

Актуальность темы. В настоящее время технология World-Wide-Web из средства, которое предоставляет графический интерфейс в Internet и упрощает доступ к информации, превращается в инструмент для распределенной работы по сети. Интенсивность освоения новых областей применения вычислительной техники и появление новых технологий приводят к необходимости совершенствования средств проектирования. Разрабатываются новые технологические решения построения приложений в среде Internet и Intranet. На сегодняшний день сложно представить процесс получения проектных решений без использования САПР и проект, который разрабатывается в одном месте и одним проектировщиком. В виду этого, учитывая сложность проектируемых устройств, большое внимание уделяется построению среды взаимодействия, которая обеспечивает совместную работу проектировщиков.

Приложения, базирующиеся на таких концепциях, называются RIA – Rich Internet application (Насыщенное интернет-приложение). Это web-приложение, доступное через Интернет, насыщенное функциональностью традиционных настольных приложений, которое предоставляется либо уникальной спецификой браузера, либо через плагин.

Для формирования и обработки VHDL структур широкое распространение получили САПР, основанные на технологии трансляции описания вычислительных устройств на языке описания аппаратуры. Созданные проектные решения на языке VHDL могут быть использованы в течение длительного промежутка времени без внесения изменений в код. Таким образом, через 5 и 10 лет найдется САПР, поддерживающая старые разработки.

Для повышения успешности реализации таких проектов применяются современные парадигмы коллективного проектирования, повторного использования кода, паттернов и др. Большое внимание уделяется инструментам организации среды взаимодействия для совместной работы, используются системы автоматизации процессов управления проектами в соответствии с требованиями PMI (Project Management Institute) и стандартами ISO. Ведутся

работы по созданию специализированных сред проектирования, позволяющих проводить распараллеливание проектных процедур, организовать взаимодействие между проектировщиками путем обмена данными и сообщениями в режиме online, быстро реализовать базовую функциональность для анализа работы системы в целом и др.

Возрастание требований к функциональным характеристикам вычислительных устройств приводит к увеличению их сложности, в частности, микросхемы СБИС являются одними из самых сложных технических объектов. Для проектирования таких устройств в настоящее время используются высокоуровневые языки описания аппаратуры HDL (Hardware Description Language): VHDL, Verilog, SystemC, System Verilog и др.

Оценка этой сложности производится с использованием метрики SLOC (Source Lines of Code), в которой подсчитывается количество строк в тексте проектируемого устройства. Данный показатель используется для прогноза трудозатрат на разработку конкретного устройства на языке описания аппаратуры, либо для оценки производительности труда уже после того, как будет создано проектное решение.

До появления системы на кристалле (SoC - System on Chip), при реализации интегральных схем, величина RTL кода была не более ста тысяч строк, а количество логических вентилей доходило до 10 млн. SoC позволила создавать целую встроенную систему на одном чипе. При проектировании интегральных схем с использованием SoC величина RTL кода может составлять более одного миллиона строк, а количество логических вентилей доходить до ста млн. [23].

При использовании традиционных методов квалифицированный разработчик может выполнять проект со средней скоростью порядка 100 вентилей в день или 30 строк RTL (Register Transfer Language) кода. В этом случае, чтобы спроектировать СБИС сложностью 100 тыс. вентилей, потребуется 1000 человеко-дней, то есть команда из пяти человек сможет разработать такую СБИС в течение года. Следуя данной логике, разработка

сложной СБИС порядка 10 млн. вентилей в течение одного года требует команду из 500 человек, что неприемлемо с точки зрения стоимости разработки [94].

Для оценки современного состояния проблемы проектирования электронных систем приведем некоторые количественные характеристики больших проектов:

1) для схем ASIC – более 20 млн. вентилей в кристалле, для ПЛИС – более 5 млн. вентилей;

2) для описания поведения проекта на системном уровне требуется более 0,5 млн. строк кода на языке C;

3) при описании проекта на уровне регистровых передач используется более 5 млн. строк кода RTL.

Примерами сложных проектов разработки аппаратных средств являются: внутрисалонная информационно-управляющая система (Cabin Intercommunication Data System, CIDS) - более чем 5 млн. строк кода, написанных, в основном, на Си, Java и VHDL [6]; описание поведенческого и структурного уровней микропроцессора Chameleon, который был разработан SGS-THOMSON Microelectronics - нескольких сотен тысяч строк [22]; кристалл SPARCv9 с тестами и окружением - около 160 тыс. строк на Verilog; большие аппаратные конструкции, которые включают несколько интегральных схем специального назначения (ASIC) и микропроцессоры, могут достигать размеров нескольких сотен тысяч строк VHDL-кода и тысяч компонентов [14].

Формирование команды разработчиков является следствием возрастающей сложности процесса проектирования высокоинтегрированных СБИС. Участники в рассматриваемых командах обладают различными знаниями и опытом в области проектирования и часто при выполнении проектов СБИС расположены на удаленном расстоянии друг от друга.

В основе организации и использования большинства HDL-языков лежит объектно-ориентированный подход. Такие языки удобно использовать как на этапах проектирования, так и верификации проекта, а созданные проектные решения могут быть использованы в течение длительного промежутка времени

без внесения изменений в код. Язык описания аппаратуры VHDL, обладая возможностью структурно-функционального представления проектируемого устройства, служит основой для формирования структурно-функциональных лингвистических моделей (СФЛМ), которые являются методологической и практической основой проектирования сложных вычислительных устройств. Под СФЛМ понимаются объекты, представленные на языке VHDL, состоящие из структурной и функциональной частей. Использование СФЛМ обеспечивает решение задач этапов проектирования вычислительных устройств по разработке интерфейсной части устройств, структуры и функционирования устройств, а также верификации.

Основными трудностями в процессе коллективного распределенного проектирования сложных HDL-проектов является организация эффективного взаимодействия проектировщиков [34] на основе современных web-технологий при решении взаимосвязанных задач проекта, управление задачами проекта, накопление базы проектных решений для ее последующего эффективного использования.

В современных системах коллективной разработки сложных программных продуктов (например, GIT, SVN, Mercurial и т.п.) используются механизмы сохранения этапов разработки, создания веток для работы над отдельными задачами с последующим слиянием в автоматическом/пользовательском режиме, ведения версий проекта с возможностью возврата к предыдущим версиям [16].
При этом для управления задачами используются ручное управление потоком задач через встроенный редактор и/или управление задачами по шаблонам бизнес-процессов в системах управления проектами. Практическое кодирование производится в системах другого типа, что затрудняет «привязку» задач к разработанному коду [76].

Указанные системы контроля версий, работая только с текстом, не могут в полной мере учесть специфику сложных HDL-проектов, заключающуюся в проектировании и верификации единой сущности «структура устройства +

функционирование устройства». HDL-языки предназначены для сквозного функционально-логического проектирования, поддерживают различные уровни абстракции проекта, включая поддержку особенностей архитектуры (architecture-specific design), являясь основным средством проектирования, моделирования и документирования от уровня вентилей до уровня цифровых систем.

Как правило, повторное использование кода при проектировании вычислительных устройств предполагает его частичное изменение, а не просто использование по принципу «As is» («Как есть»). Процесс написания reusable (многократно используемого) кода очень важен. Использование готового кода, который является работоспособным и был хорошо протестирован, оказывается экономически более выгодным в 90% случаев для компаний производителей. В случае использования готового модуля сторонний разработчик может вообще не знать о внутреннем устройстве, при этом ему важен лишь интерфейс и принцип работы.

В настоящее время отсутствуют средства обучения САПР, позволяющих обеспечить процесс передачи экспертных знаний о методиках проектирования либо напрямую от эксперта, либо на основе ряда примеров. Обеспечение такой возможности позволит улучшить эффективность использования САПР на базе языка HDL за счет повышения уровня автоматизации процесса проектирования [83].

Таким образом, в области автоматизированного проектирования сложных HDL-проектов **актуальной и имеющей большое практическое значение научно-технической задачей** является разработка методов и средств распределенного проектирования СФЛМ, позволяющих организовать взаимодействие между проектировщиками, которые находятся на удаленном расстоянии друг от друга, для решения проектных задач и получения проектных решений.

Разрабатываемые методы должны обеспечить возможность распределенного формирования проектных решений коллективом

проектировщиков путем их создания как с «нуля» в виде VHDL кода, так и из шаблонов СФЛМ с частичной доработкой. В процессе формирования проектного решения применение методов управления проектными задачами, поиска и синтеза проектных решений должны способствовать оптимизации управления сложными процессами проектирования VHDL-программ, сократить затраты разработки и повысить качество проектирования.

Степень научной разработанности проблемы. В настоящее время ведущие разработчики САПР работают в направлениях создания специализированных сред проектирования позволяющих: проводить распараллеливание проектных процедур; организовывать взаимодействие между проектировщиками путем обмена данными и общением в режиме online; быстро реализовывать базовую функциональность для анализа работы системы в целом и др.

Для разработки распределенной коллективной среды проектирования особое внимание уделяется актуальным в наше время научным работам, направленным на исследования современных достижений в техническом, математическом, информационном, методическом и организационном обеспечении САПР.

Необходимо отметить ряд работ авторов, которые в своих исследованиях затрагивают области, рассматриваемые в данной диссертации.

В работах Камаева В.А., Глушаня В.М., Лаврика П.В. [99,81,82,83,84] рассматриваются вопросы по совершенствованию эффективности САПР за счет повышения их быстродействия путем использованию сетевых информационных технологий.

В работах Анисимова В.И. [47,48,49,50] рассматриваются вопросы, связанные с построением распределенных САПР и применением интернет-технологий.

В работах Гридина В.Н., Дмитревича Г.Д., Анисимова Д.А. [51,52,53,54,55,56,114] рассматриваются вопросы, связанные с применением web-технологий при построении САПР.

В работах Ильичева Н.Б., Пантелеева Е.Р., Пекунова В.В., Первовского М.А., Целищева Е.С. [153] рассматриваются вопросы, связанные с распределенным проектированием в САПР AutomatiCS.

В работах Адамова А.З. [46,81] рассматриваются вопросы, связанные с исследованием распределенных web-ориентированных архитектур САПР.

В работах Indrusiak L.S. [31] рассматриваются вопросы, связанные с организацией совместной работы при поддержке распределенных сред разработки.

В работах Udwadia F. [36] рассматриваются вопросы, связанные с рассмотрением методологии анализа совместного процесса проектирования и конфликтов, связанных с техническими и социальными факторами.

В работах Törlind P. [37] рассматриваются вопросы, связанные с взаимодействием распределенной команды проектировщиков.

Целью диссертационной работы являются сокращение затрат на разработку и повышение качества формирования проектных решений путем реализации методов и средств коллективного распределенного проектирования, позволяющих управлять процессом проектирования и создавать базу проектных решений, элементами которой являются СФЛМ, для их повторного использования.

В соответствии с поставленной целью в работе формулируются и решаются следующие **задачи исследования**.

1. Анализ методологий, методов и средств построения распределенных и многоагентных систем коллективного проектирования.

2. Разработка многоагентной системы коллективного распределенного проектирования на основе СФЛМ.

3. Разработка модели системы распределенного проектирования (СРП) СФЛМ на базе цветных сетей Петри и проведение анализа на ее основе с целью верификации СРП, проверки сценарного взаимодействия агентов СРП.

4. Разработка методов управления проектными задачами и процессом получения проектных решений.

5. Разработка методов формирования, синтеза и поиска СФЛМ в СРП.

6. Разработка программно-информационного обеспечения СРП.

Объектом исследования является автоматизация процесса формирования проектного решения при распределенной работе нескольких проектировщиков с проектными задачами.

Предметом исследования являются модели, методы и средства распределенного проектирования, используемые для организации коллективной работы и обеспечивающие автоматизацию процесса формирования проектного решения, представленного в виде СФЛМ.

Методы исследования основаны на использовании теории многоагентных систем, теории графов, теории сетей Петри, теории построения, web-ориентированных САПР, теории баз данных.

Научная новизна полученных в диссертации результатов теоретических и экспериментальных исследований определяется разработанными методами и средствами, лежащими в основе организации и функционирования СРП СФЛМ. В результате исследований получены следующие результаты.

1. Предложена новая архитектура многоагентной системы СРП, включающая структуру подсистем и агентов, отличающаяся составом, типами и функциональностью агентов, и позволяющая обеспечить коллективное распределенное проектирование сложных VHDL-объектов.

2. Предложен новый метод управления задачами в СРП, отличающийся использованием разработанной ассоциативно-ориентированной параллельной сетевой схемой задач (ПССЗ) и позволяющий оптимально организовать выполнение проектных задач.

3. Предложен новый метод формирования библиотек VHDL-программ, которые позволяют многократно использовать проектные решения или модифицировать их с учетом новых задач, применяя концепцию повторного использования (Reuse). Данный метод, в отличие от существующих, позволяет

наполнять библиотеку СФЛМ, которые были созданы с применением концепции MVC для разделения интерфейсов, описаний функционирования, структур проектируемых устройств и представлений.

Практическая ценность

Практическими результатами диссертационной работы являются.

1) Разработана модель архитектуры системы распределенного проектирования на базе цветной сети Петри, позволяющая провести имитационное моделирование с целью анализа и верификации нового функционала при масштабировании системы или внесении изменений в существующий функционал.

2) Разработано программное обеспечение web-ориентированной многоагентной СРП.

3) Разработан web-ориентированный транслятор, позволяющий обнаруживать ошибки в описании на VHDL при проектировании устройства и формировать СФЛМ из этого описания.

4) Разработан распределенный поиск проектных решений, представленных
в виде СФЛМ.

5) Разработан аппарат управления процессом распределенного проектирования путем формирования операторов в ПССЗ.

На защиту выносятся следующие новые и содержащие элементы новизны основные результаты.

1. Организация распределенного проектирования сложных VHDL-объектов в виде многоагентной системы, содержащей 8 типов агентов с их ролевой функциональностью;

2. Механизмы управления процессом проектирования через аппарат ПССЗ;

3. Применение концепции MVC на этапе формирования проектируемого устройства из совокупности СФЛМ.

4. Разработанное программно-информационное обеспечение СРП.

Реализация и внедрение. Результаты работы внедрены в АО «Ульяновский механический завод» путем использования в корпоративной компьютерной среде обучения предприятия, в ООО «Разработка кибернетических систем» путем использования в процессе разработки проектов, а также в учебный процесс кафедры «Вычислительная техника» УлГТУ.

Апробация работы. Основные положения диссертационной работы докладывались и обсуждались на следующих Международных и Всероссийских конференциях: VIII Международной научно-практической конференции «Объектные Системы-2014», г. Шахты, 2014; VIII Международной научно-практической конференции «Системы проектирования, моделирования, подготовки производства и управление проектами CAD/CAM/CAE/PDM», г. Пенза, 2014; Всероссийской научно-технической конференции «Информатика и вычислительная техника», г. Ульяновск, 2014; Всероссийской школе-семинаре «Информатика, моделирование, автоматизация проектирования», г. Ульяновск, 2013-2014; Конгрессе по интеллектуальным системам и информационным технологиям «IS&IT'14», п. Дивноморское, 2014; XXXVII Международной научно-практической конференции «Технические науки – от теории к практике», г. Новосибирск, 2014.

Публикации. По теме диссертации опубликованы 15 печатных работ, в том числе 5 статей в изданиях, входящих в перечень ВАК РФ. Получено 1 СВИДЕТЕЛЬСТВО (РОСПАТЕНТ) о государственной регистрации программы для ЭВМ №2015610356.

ГЛАВА 1. МЕТОДОЛОГИИ, МЕТОДЫ И СРЕДСТВА РАСПРЕДЕЛЕННОГО ПРОЕКТИРОВАНИЯ СФЛМ

Целью данной главы является исследование методологий, методов и средств организации распределенного взаимодействия между проектировщиками в процессе проектирования. Рассматривается СФЛМ, как основной объект, в проектировании ВУ. Проведен анализ современных сред проектирования на языках описания аппаратуры HDL и их возможностей для организации распределенного, коллективного проектирования.

1.1. Место и роль структурно-функциональных лингвистических моделей в проектировании ВУ

В связи с быстрым ростом степени интеграции и функциональной сложности современных электронных устройств возникает необходимость совершенствования и развития методов проектирования больших и сверхбольших интегральных схем (БИС и СБИС). Многоуровневое иерархическое представление устройств выполняется по методологии нисходящего проектирования (сверху вниз). По данной методологии разработка ведется от общего описания системы к детальному описанию ее функциональных модулей.

При разработке уделяется внимание поведенческому и функциональному представлению системы, не отвлекаясь на структуру. Описание функционирования аппаратуры и представление структурно-функциональной лингвистической модели (СФЛМ) осуществляется с помощью HDL (Hardware Description Language) языков описания. Они представляют собой набор синтаксических и семантических правил, определяющих формальную запись, которая используется на всех этапах проектирования цифровых электронных систем.

Применяемые при проектировании языки делятся три группы – описательные (структурные), диалоговые (директивные) и моделирующие (процедурные).

Основными частями языка описания являются: описание объекта; описание задачи; описание директив проектирования. При описании, объект представляется совокупностью отдельных элементов, каждый из которых состоит из множества блоков, которыми могут являться: тип элемента, тип модели элемента, параметры модели элемента, технологические связи элемента.

Языки моделирования (процедурные языки) позволяют описать структуру, параметры объекта проектирования и алгоритм его функционирования. В качестве примера можно представить процесс передачи и преобразования сигнала от блока к блоку.

Существует несколько разновидностей данных языков: AHDL, VHDL, VerilogHDL, Abel и др. Иногда используются стандартные языки программирования, например, язык Си для описания структуры БИС.

Язык VHDL (VHSIC - Very High Speed Integrated Circuits – Hardware Description Language) изначально предназначался для описания аппаратуры и обеспечения обмена проектами между различными соисполнителями работ по созданию высокоскоростных интегральных схем. Язык позволяет проводить моделирование на вентильном уровне, уровне регистровых передач и корпусов микросхем, а также осуществлять синтез устройств.

Проводится организация цифровой системы на языке VHDL [33,77,98] и описываются функции, определяющие преобразование значений на входах в значения на выходах. Таким образом организация системы задается перечнем связанных компонентов.

Проект на языке VHDL представляет собой совокупность файлов. В файлах содержится последовательность лексических элементов, каждый из которых составлен из строго установленного набора символов. Лексический элемент может быть представлен в виде ограничителя, идентификатора (являющегося служебным словом), абстрактного, символьного, строкового или битово-строкового литерала, либо комментария.

Различные стили описания аппаратных архитектур представлена на рис. 1.1.

1. Структурное описание представляется в виде иерархии связанных компонентов.

2. Потоковое описание (описание данных) представляется в виде множества параллельных регистровых операций, каждая из которых управляется вентильными сигналами.

3. Поведенческое описание описывается последовательными программными предложениями, похожими на имеющиеся в любом современном языке программирования высокого уровня предложениями.

Все три стиля могут совместно использоваться в одной архитектуре [98].

Для проектирования цифровых схем используются структурное и потоковое описания, а поведенческое – только для моделирования, в виду того, что содержит конструкции, не реализуемые в виде схемы.

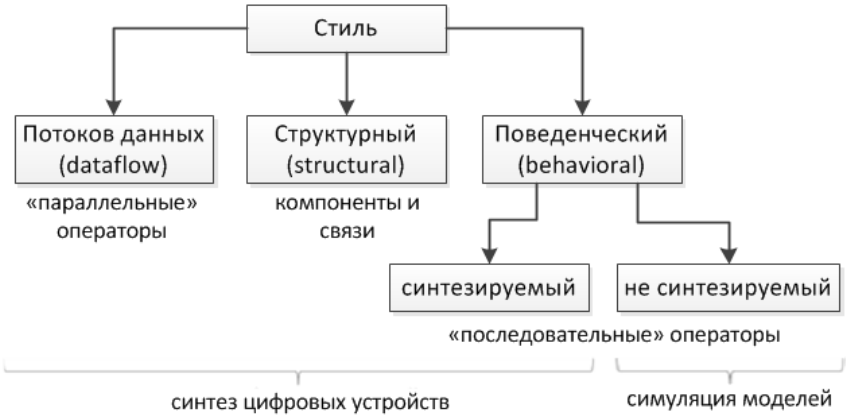


Рис. 1.1. Схема стилей описания аппаратных архитектур

Опишем основные структурно-функциональные сущности языка VHDL:

1) Объект проекта (*entity*) описывает компонент проекта, содержащий заданные входы и выходы и осуществляющий определенные действия. В качестве объекта может быть представлена как вся проектируемая система, так и определенная подсистема, устройство, узел, стойку, плату, кристалл, макроячейку, логический элемент и т. п. Описание объекта может быть построена в виде совокупности компонентов, которые, были реализованы ранее в виде самостоятельных объектов;

2) Объявление объекта проекта (*entity declaration*) описывает интерфейс и определяет только входы и выходы проектируемого объекта;

3) Архитектурное тело (*architecture body*) описывает поведение объекта и его структуры;

4) Объявление конфигурации (*configuration declaration*) необходимо для определения, какие модули должны быть использованы при создании всего проекта.

Активное применение VHDL началось при проектировании устройств на программируемых интегральных микросхемах (ПЛИС). На рынке ПЛИС представлены микросхемы FPGA- и CPLD-структур. FPGA (Field Programmable Gate Array) представляет собой микросхему, которая может быть сконфигурирована самим проектировщиком. Определенные ячейки которой, отвечают за реализацию элементарных функций (Configurable Logic Blocks CLB), а другие – за внутренние соединения (Programmable Switch Matrices PSM). Многоступенчатый процесс проектирования устройства представлен на рис. 1.2.

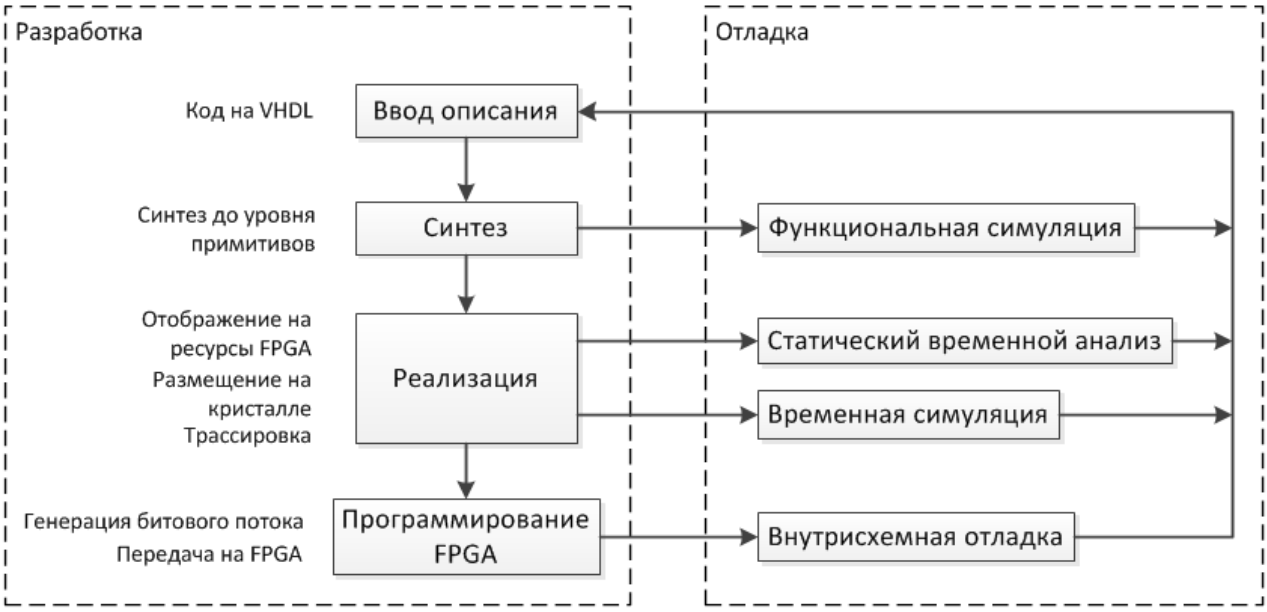


Рис. 1.2. Процесс проектирования устройства

При разработке вычислительных систем всех уровней сложности может быть применен язык VHDL. Разделяя проект на задачи или этапы, можно организовать процесс проектирования путем делегирования задач нескольким группам разработчиков. В качестве коммерческого результата проекта используется поведенческое и потоковое описание устройства. Данные

переданные заказчику, впоследствии могут быть использованы для дальнейшей доработки проекта.

СФЛМ представляет собой совокупность объектов и связей между ними. Структура СФЛМ представлена на рис. 1.3. В описываемой структуре встречаются квантификаторы, которые показывают, сколько раз могут встречаться элементы в описываемой структуре. Например, (*) предполагает неограниченное количество элементов, а (+) хотя бы один элемент. Обрабатываемые данные хранятся в входных и выходных сигналах, ячейках памяти, сигналах связи и регистрах. При поступлении данных на входы обрабатываемого объекта в соответствии с функцией преобразования происходит формирование выходных результатов. При формировании шаблона из описанной модели СФЛМ можно управлять настраиваемыми параметрами и, таким образом, получить новое проектное решение. На основе описанной структуры организуется поиск СФЛМ по параметрам, которые являются важными атрибутами модели [149].

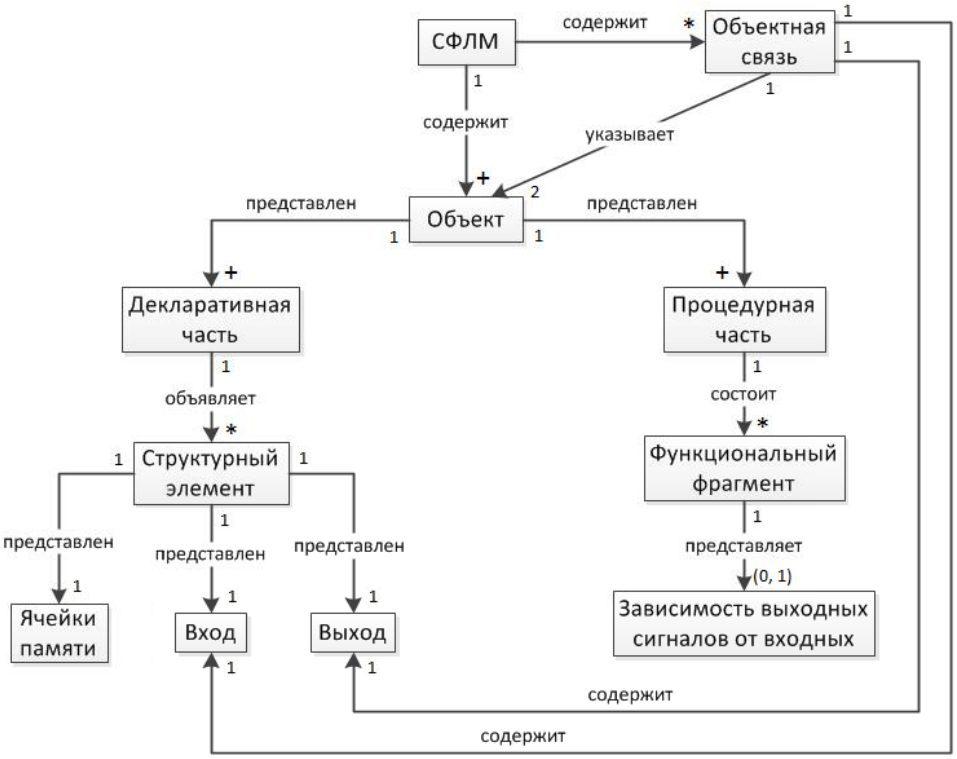


Рис. 1.3. Структура СФЛМ

В ходе процесса проектирования формируется шаблон СФЛМ. Особенность шаблона состоит в том, что при изменении параметров можно получить новое проектное решение. Размерности ячеек памяти, отвечающих за структурные элементы объектов и связи между ними, рассчитываются относительно предварительно указанных входных переменных шаблона.

Сравним различные виды описания функционирования проектируемого устройства [110]. Результат сравнения сведем в табл. 1.1.

Таблица 1.1. Сравнение видов описания функционирования проектируемого устройства

Показатель	**Текстовое описание на HDL**	**Графическое описание**	**Табличное описание**	**СФЛМ**
Сложность и скорость создания проекта	Повышение скорости	Повышение скорости для проектов небольшой сложности	Повышение скорости	Повышение скорости
Наглядность и компактность	Да	Для схем небольших размеров	Частичная аналогия с документацией на устройство [111]	Да
Отладка и верификация	Да	Частично	Нет	Не требуется
Возможность повторного использования	Да	Да	Да	Да
Синтез проекта	Да	Нет	Нет	Нет
Наличие ошибок	Да	Да	Да	Нет
Использование в качестве элемента вычислительной заготовки (IP Core) [156]	Нет	Нет	Нет	Да
Требования к квалификации проектировщика	Знание языка описания аппаратуры и области проектирования	Знание графической системы и области	Знание языка описания аппаратуры и	Знание в области проектирования

		проектирования	области проектирования	
Гибкость и функциональность описания устройств	Да	Нет	Да	Да
Форма представления	Одномерная	Двухмерная трехмерная	Двухмерная	Двухмерная
Ограниченность возможностей и области применения в проектировании	Нет	Да	Введено ограничение варьируемости, с целью повышения надежности [110]	Нет

Наиболее распространенными видами описания функционирования проектируемого устройства являются текстовое и графическое, но каждое из них имеет ряд недостатков, которые могут быть устранены при использовании совместно с СФЛМ.

1.2. Анализ недостатков применяемых средств создания проектных решений

К типу EDA (electronic design automation) или ECAD (electronic computer-aided design) относятся множество систем, представляющих собой САПР электронных устройств, радиоэлектронных средств, интегральных схем, печатных плат и т. п. [90,91,93].

В ходе анализа существующих САПР различного типа были выявлены основные возможности, позволяющие организовать процесс распределенного проектирования.

1. Применение проектного хранилища (Design Repository) представляющего собой централизованное хранилище, в котором содержатся все проекты. Проектное хранилище является основным инструментом для

проектировщиков, оно принадлежит проектным отделам и содержит подробную историю процесса проектирования.

2. Использование систем контроля версий, таких как CVS (Concurrent Version System), SVN, GIT, Mercurial для обмена проектными решениями с целью получения актуальных данных в процессе коллективной разработки. Данное решение направлено на отслеживание состава всех файлов проекта, создаваемых в САПР, на предмет их целостности, непротиворечивости и актуальности.

3. Применение многопользовательской параллельной методологии разработки для ускорения процесса проектирования. Организуется доступ к общей базе данных, за счет этого разработчики могут работать одновременно, разделяя процесс проектирования на ряд задач или областей. Основными типами являются: разделение проекта по слоям, функциональным блокам, цепям и уровню доступа.

4. Применение технологии Internet Team Design (Itd) с целью поддержки коллектива разработчиков, как в локальной сети, так и с использованием ресурсов Internet.

5. Представление актуальной проектной информации, с уведомлением о необходимости обновления измененных данных.

6. Подключение к компьютеру, находящемуся в одной сети, для поиска нужного схемного проекта, и последующей работы с ним.

7. Информирование членов проектной команды о проектных задачах.

8. Обмен информацией между членами проектной команды [88], исключающий повторного ввода данных, быстрого внесения исправления в проект, отслеживания работы других сотрудников в режиме реального времени.

Из существующих систем можно выделить несколько групп, которые необходимо рассмотреть для выявления достоинств и недостатков, связанных с процессом распределенного проектирования, формирования проектных решений и управления проектными задачами.

1. Данная группа представляет САПР, устанавливаемых на ПК пользователя с лицензией на использование. Изменение функционала подобной САПР выполняется разработчиками путем выпуска новых версий продукта. Очевидный недостаток подобной САПР – отсутствие интеграции со сторонними системами. Такие типы САПР позволяют получать проектные решения в процессе коллективного проектирования путем обмена данными по почте и др. средствами связи. Внутренняя структура интересующих функциональных блоков представлена на рис. 1.4. К этой группе относятся такие САПР как QuartusII, Xilinx Foundation Express, WebPACK ISE, FOUNDATION, HDL Designer Mentor Graphics, MAX+PLUS II, Synopsys, и др.

Рис. 1.4. Структура первой группы САПР

2. Данная группа обладает возможностями предыдущей группы. Дополнительно вводится функционал по интеграции со сторонними программами [27] (включая САПР). Взаимодействие в подобных САПР происходит посредствам установки плагинов непосредственно в систему или написанием сторонней программы, которая будет формировать или запрашивать данные от САПР через API. Коллективное проектирование в подобных САПР возможно посредствам плагинов, организующих подобное взаимодействие. Внутренняя структура интересующих функциональных блоков представлена на рис. 1.5. К этой группе относятся такие САПР как ModelSim и Altium Designer.

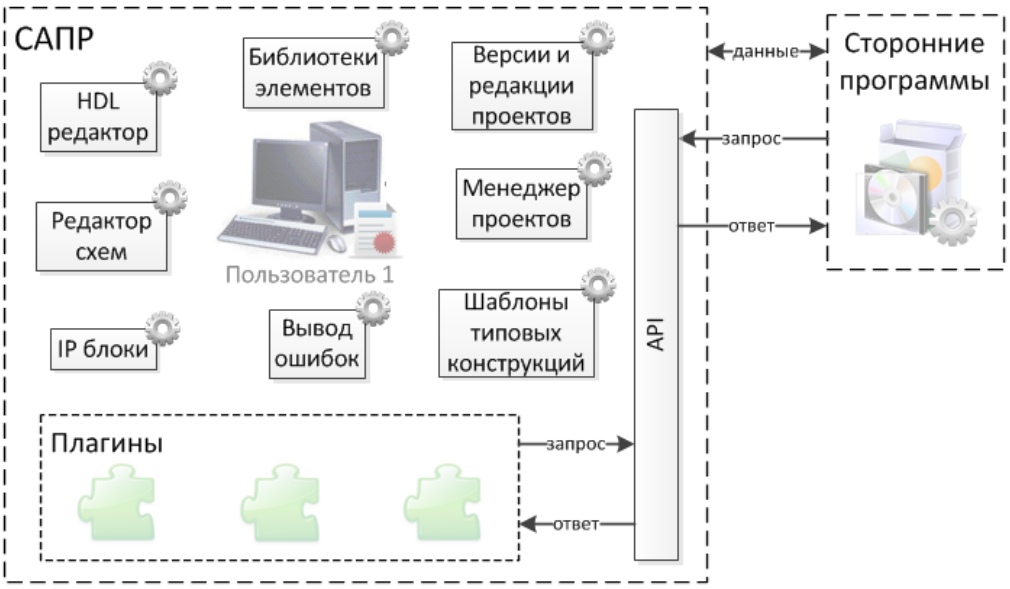


Рис. 1.5. Структура второй группы САПР

3. Данная группа обладает возможностями первой и зачастую возможностями второй группы. Дополнительно вводится функционал контроля версий, позволяющий организовать коллективное проектирование. Данный вид САПР организует интерактивное взаимодействие между проектировщиками путем обмена проектными решениями через общий репозиторий. Внутренняя структура интересующих функциональных блоков представлена на рис. 1.6. К этой группе относятся такие САПР как Altium Designer [33,100].

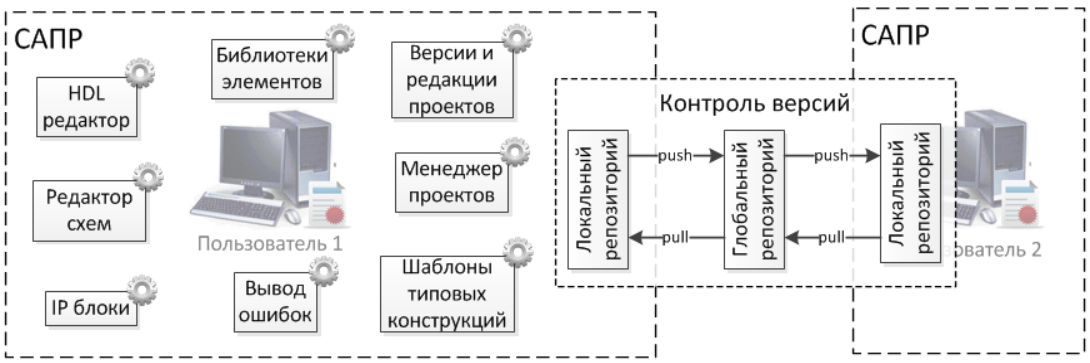


Рис. 1.6. Структура третьей группы САПР

В ходе анализа были выявлены определенные функциональные особенности, способствующие распределенному проектированию и формированию проектных решений. Данные особенности частично присутствуют в САПР, тип которых отличен от EDA / ECAD. На рис. 1.7 представлены эти особенности в виде доп. функциональных блоков.

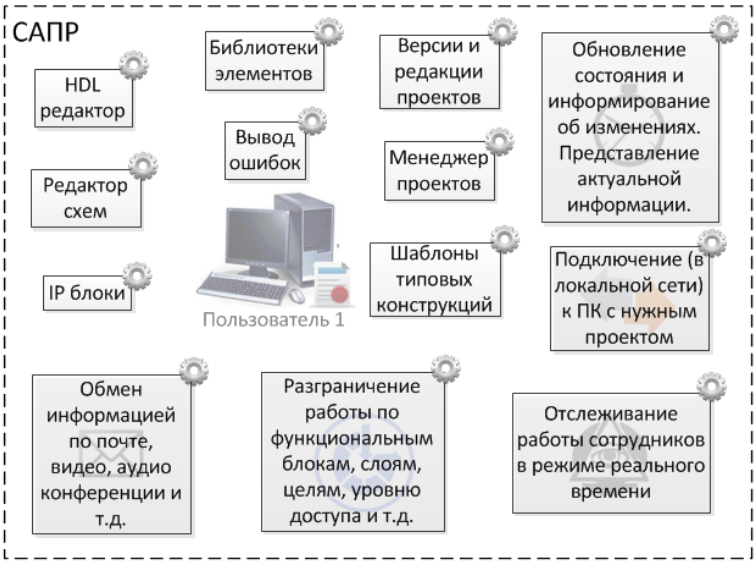


Рис. 1.7. Функциональные блоки распределенного проектирования в САПР

В результате анализа представим архитектуру распределенной САПР в виде функциональных блоков. При построении данной архитектуры были учтены положительные и отрицательные особенности построения, функционирования и поддержки, существующих САПР как для распределенного получения проектных решений, так и организации процесса проектирования, учитывая отсутствие привязки проектировщика к рабочему месту. Архитектура в виде функциональных блоков СРП представлена на рис. 1.8.

Рис. 1.8. Архитектура СРП в виде функциональных блоков

Сравним функционал получения проектных решений в различных ECAD системах. Результат сравнения сведем в табл. 1.2.

Таблица 1.2. Сравнение функционала получения проектных решений в ECAD системах

<table>
<tr><th>Показатель</th><th>Altium Designer</th><th>Active-HDL</th><th>Mentor Graphics HDL Designer</th><th>OrCAD</th><th>P-CAD</th><th>ModelSim</th></tr>
<tr><td>Коллективная работа</td><td>Да</td><td>Нет</td><td>Нет</td><td>Да</td><td>Да</td><td>Нет</td></tr>
<tr><td rowspan="2">Способ получения проектного решения</td><td colspan="6">Текстовый HDL редактор</td></tr>
<tr><td colspan="5">Редактор схем</td><td></td></tr>
<tr><td rowspan="4">Наличие доп. помощников при получения проектного решения</td><td colspan="6">Библиотека элементов</td></tr>
<tr><td colspan="3">IP блоки</td><td rowspan="3">Шаблоны типовых конструкций, Internet Component Assistant</td><td rowspan="3">Создание иерархических проектов</td><td rowspan="3">Работа с сохраненными сценариями моделирования в виде пакетных командных файлов</td></tr>
<tr><td colspan="2">Language Assistant, шаблоны типовых конструкций</td><td rowspan="2">Версии и редакции проекта</td></tr>
<tr><td>Design Repository, Version Control</td><td></td></tr>
<tr><td>Область применения</td><td>Реализация проектов в электронных средствах на уровне схемы или программного кода</td><td>Моделирования и верификации проектов для ПЛИС</td><td>Автоматизация проектирования электронных устройств (EDA)</td><td>Создание электронных версий печатных плат для производства плат и электронных схем Моделирование схем</td><td>Проектирование многослойных печатных плат вычислительных и радиоэлектронных</td><td>Моделирование и отладка программ на HDL языках</td></tr>
</table>

					устройств	
Поддержка	Да	Да	Да	Да	Нет	Да
Кросплатформенность	Нет	Нет	Нет	Нет	Нет	Нет
Возможность интеграции	Да	Нет	Нет	Нет	Нет	Да
Лицензия	Под конкретный ПК					
	Привязка к аккаунту AltiumLive, на Private License Server					

Наиболее оснащенной функционалом для получения проектных решений является ECAD система Altium Designer, но в ней есть ряд недостатков связанных с кроссплатформенностью, лицензированием и использованием существующих возможностей web-технологий для организации распределенного проектирования.

1.3. Анализ средств распределенного взаимодействия между проектировщиками

Под средствами распределенного взаимодействия между проектировщиками будем понимать системы, обладающие достаточным функционалом для решения задач коллективного проектирования.

Применение системы контроля версий типа Git для организации распределенного проектирования связано с созданием сервера разработки для синхронизации проектных решений. Ввиду того, что каждый проект состоит из большого числа вносимых изменений, разными проектировщиками. Система контроля версий (СКВ) позволяет следить за этими изменениями и видеть прогресс разработки проектируемого устройства. Это делает СКВ важным инструментом в повседневной работе проектировщика, работающего в команде.

Рассмотрим реализацию web-систем, представляющих собой клиент-серверное приложение, в котором клиентом выступает браузер, а сервером – веб-сервер. Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по сети. Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому веб-приложения являются кросплатформенными.

Для организации взаимодействия между распределенными клиент-серверными системами применяется технология Common Object Request Broker Architecture (CORBA). В этой модели реализация объектов происходит в адресном пространстве сервера. Создание клиент-серверной СРП происходит в соответствии с правилами CORBA. Отличительным признаком CORBA является архитектура Object Management Architecture (OMA) [112]. В основе OMA лежит концепция брокера объектных запросов. Брокер объектных запросов (ORB, Object Request Broker) – основной компонент CORBA [152], относящийся к классу middleware, т.е. связующему программному обеспечению. Объекты, которые используют брокер объектных запросов, могут взаимодействовать с любыми другими объектами, подключенными через брокер.

Все системы в распределенной среде рассматриваются как объекты, которые могут одновременно играть роль и клиента, и сервера. Если объект является инициатором вызова метода у другого объекта, то он выполняет роль клиента. Если объект является исполнителем запрашиваемого метода, то он выполняет роль сервера. Большинство объектов одновременно исполняют роль и клиентов, и серверов, попеременно вызывая и исполняя методы. Использование CORBA позволяет строить гораздо более гибкие системы, чем системы клиент-сервер, основанные на двухуровневой и трехуровневой архитектуре.

Для достижения распределенного взаимодействия между проектировщиками различных подразделений одной организации, применяется частное облако, как средство организации процесса проектирования с

централизованным хранением проектных решений. Под частным облаком будем понимать инфраструктуру, предназначенную для использования одной организацией, включающей несколько подразделений.

Проанализировав существующие тенденции перехода САПР в облака, были выделены 3 уровня в процессе перехода к облачным технологиям.

1. САПР остается привязанной к локальному ПК проектировщика, в виду сформировавшихся достаточно емких библиотек элементов, хорошо проработанной концепции получения проектного решения в desktop системе, несмотря на отсутствие или слабом развитии средств коллективной разработки, и наличия вопросов о безопасности хранимых в облаке данных. К подобным САПР относятся EDA / ECAD системы.

2. На данном уровне выделяются 3 типа систем.

a) системы управления инженерными данными (хранилище), к которому можно получить доступ из любой точки мира при наличии сети Internet.

b) САПР или ее функциональная часть разрабатывается в виде web-приложения [148], которое позволяет получать проектные решения, используя web-браузер Примером подобных системы являются PCBWeb, Wedana и др.

c) Создается web-ресурс в виде совокупности проектных решений (заготовок), которые можно скачать и интегрировать в разрабатываемый проект. Подобным ресурсом является opencores.org.

3. САПР имеет не только desktop, но и облачную версию. Зачастую облачные версии обладают меньшей функциональностью, но большей производительностью по сравнению с мощностью ПК пользователя.

Описанная трехуровневая тенденция перехода САПР в облака представлена на рис. 1.9.

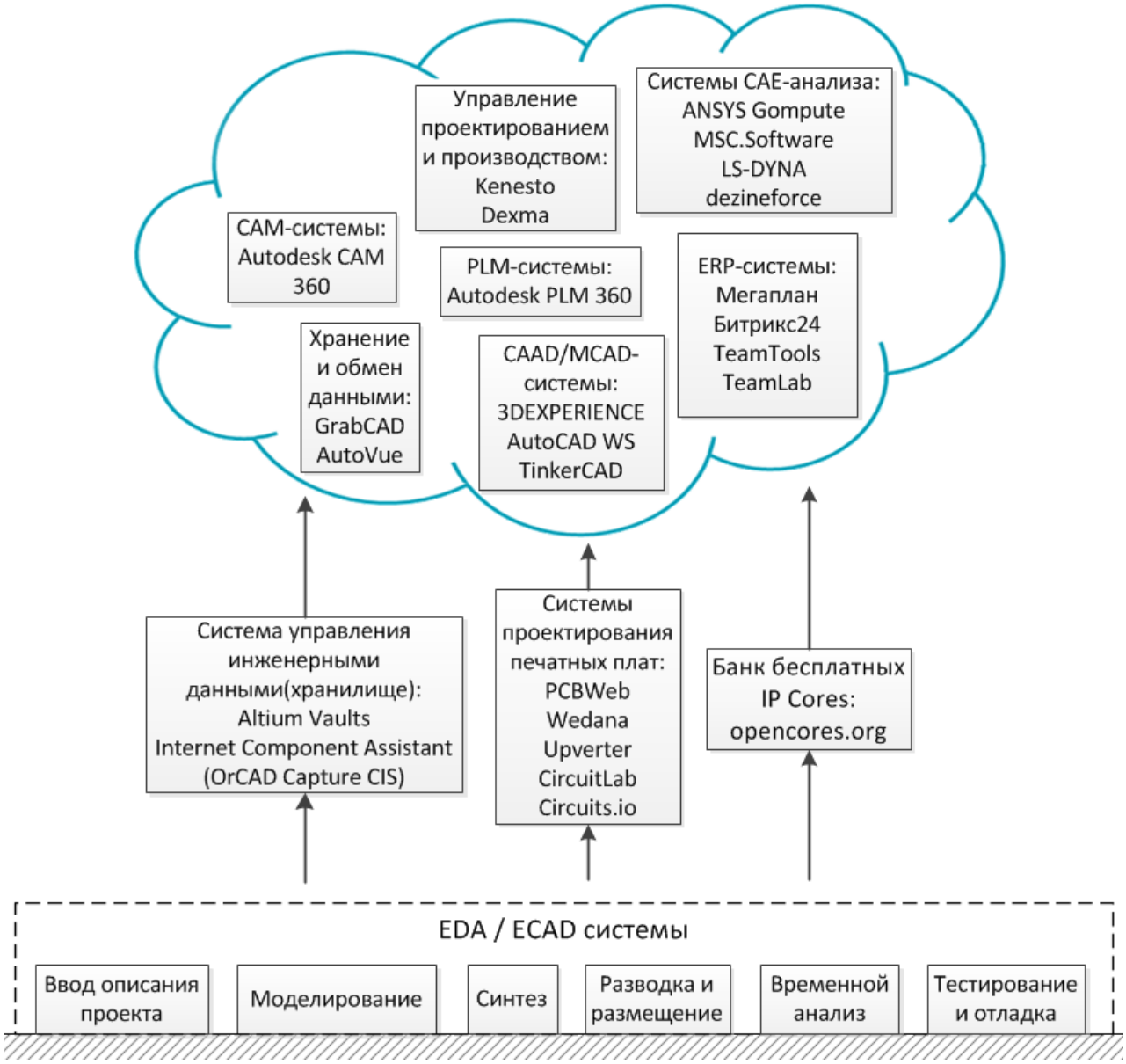


Рис. 1.9. Трехуровневая тенденция перехода САПР в облака

Сравним средства позволяющие организовать распределенное взаимодействие между проектировщиками. Результат сравнения сведем в табл. 1.3

Таблица 1.3. Сравнение средств распределенного взаимодействия между проектировщиками

Показатель	Web-система	Частное облако	CORBA
Декомпозиция задач проектирования	Web-интерфейс и/или доп. интеллектуальный функционал web-системы. Наличие централизованного хранилища задач	Использование функционала по модели SaaS через web-интерфейс	Сервер централизованного хранилища задач с описанием интерфейса взаимодействия на языке idl

Распределение проектных задач	Web-интерфейс и/или доп. интеллектуальный функционал web-системы. Наличие централизованного хранилища задач	Использование функционала по модели SaaS через web-интерфейс.	Сервер централизованного хранилища задач, функционал для клиента и сервера с описанием интерфейса взаимодействия на языке idl
Управление изменениями в проекте	Интеграция с Git и/или доп. функционал web-системы	Интеграция с Git и/или доп. функционал системы в виде модели SaaS	Интеграция с Git и/или доп. функционал системы с описанием интерфейса взаимодействия на языке idl
Организация взаимодействия между проектировщиками	Web-интерфейс, почта	Использование функционала по модели SaaS через web-интерфейс	Взаимодействие между клиентом и сервером по протоколу IIOP/GIOP

Наиболее применяемым средством коллективного взаимодействия является web-система, в виду своей простоты разработки и внедрения. Однако при создании инфраструктуры, предназначенной для использования одной организацией, включающей несколько подразделений, больше подходит частное облако. Несмотря на сложность процесса разработки, в основе которой лежит объектно-ориентированные модели, CORBA позволяет разрабатывать распределенные приложения и организовывать интеграцию распределенных изолированных систем за счет создания универсального описания интерфейса на языке IDL.

1.4. Методологии распределенного проектирования объектов СВТ

Рассматриваются методологии, составляющие основу для проектирования и разработки СРП. Каждая из них задает определенную последовательность проектных процедур, выполнение которых обеспечивает организацию процесса

коллективного взаимодействия. Данные методологии разработки благодаря ясному общему представлению помогают охватить все важные шаги процесса распределенного проектирования.

Workflow методология

С помощью методологии семейства IDEF (Integrated DEFinition) [5] осуществляется моделирование систем управления процессами проектирования. Для описания и документирования информационных потоков системы, процессы в которой выполняются в заданной последовательности, используется методология графического моделирования IDEF3 (workflow diagramming).

Workflow фиксирует процесс и включая правила управляющие его выполнением (графики, приоритеты, маршруты, авторизация, уровень секретности и роль каждого участника процесса).

Любое сложное проектное решение представляет собой совокупность результатов работы специалистов в области проектирования разных квалификаций (инженеров-системотехников, схемотехников, разработчиков топологии, инженеров по верификации и инженеров-тестировщиков). Применение методологии позволяет следить за процессом проектирования, и в позволяет отслеживать где именно процесс проектирования пошел по неправильному пути, изменить его и сделать более эффективным.

При коллективной разработке, функциональные блоки, включающие RTL и соответствующие схемные ограничения должны поддерживать повторное использование модулей или всего проекта.

На данный момент существуют инструменты, позволяющие выполнять работу и управлять ее процессом. В подобных системах workflow, после формального задания производственного процесса, осуществляется управление ходом работы через специальную программу. В ходе работы формируются и назначаются, делегируются задачи и отслеживается прогресс выполнения [79].

Формальное представление процесса, включает в себя:

1) представление элементарных операций, формирующие процесс;

2) представление исполнителей, выполняющие указанные операции;

3) представление зависимостей между операциями;

4) представление внешних событий, влияющих на ход процесса, и правил их обработки.

Базовые понятия технологии workflow [79] представлены на рис. 1.10. в виде понятийной информационной модели.

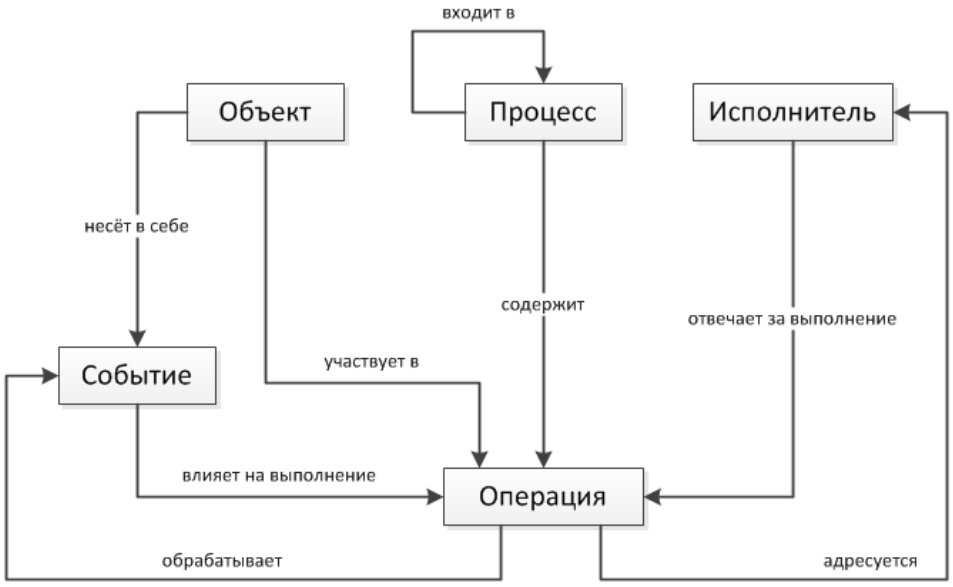


Рис. 1.10. Понятийная информационная модель технологии workflow

Эта информационная модель подходит для распределенного проектирования. Объектами в такой модели могут быть: код на VHDL, шаблоны СФЛМ, проектные задачи, проектные решения. Уведомление о постановке проектной задачи, окончании создания проектного решения, запросе поиска шаблона СФЛМ в базе данных и др. выступают в качестве событий в workflow модели. Исполнителями являются проектировщики, архитекторы. В качестве операций используются основные функции системы, такие как поиск шаблона СФЛМ в базе данных, запрос на поиск данных на удаленном сервере, создание шаблона СФЛМ путем анализа описания на VHDL, постановка проектных задач и назначение задач проектировщику.

Методология сопряженного проектирования

Сопряженное проектирование представляет собой процесс параллельного проектирования аппаратных и программных средств при котором происходит оценка целесообразности выбора аппаратной или программной реализации

определенной части проекта. В ходе этого процесса создаются наиболее эффективные конфигурации, что приводит к сокращению времени разработки и позволяет проектировщикам увидеть работу разрабатываемой системы с разделением аппаратных средств и программного обеспечения. Сопряженное проектирование рассматривалась в следующих источниках [91, 113, 117].

Процесс сопряженного проектирования не разделим с процессом сопряженной верификации совместной работы программного обеспечения и аппаратных средств системы с целью определения их корректного совместного функционирования. Задача современного сопряженного проектирования системы – сокращение временных затрат и уменьшение числа итераций. Основные этапы процедуры сопряженного проектирования, базирующиеся на идеях сопряженной верификации.

Т.к. система на кристалле – изделие не только очень сложное, но и дорогое при небольших объемах выпуска, ошибки проектирования должны быть исключены. Поэтому в данной методологии, кроме математического моделирования, предусматривается этап создания прототипа устройства на базе специальных аппаратно-программных платформ, которые сегодня называются – алгоритмически ориентированные платформы проектирования. Они содержат набор элементов: ПЛИС, процессорные ядра, память, определенную шинную архитектуру, интерфейсы и т.д., которые позволяют создать прототип схемы до ее воплощения в СБИС.

Концепция сопряженного проектирования предполагает следующее:

1) Проведение анализа и декомпозиция задач на фрагменты, в зависимости от имеющихся ресурсов, которые будут реализованы аппаратно и/или программно. Таким образом, максимизировать показатель качества системы в целом. Формализация процедуры предварительного распределения фрагментов является сложной задачей, поэтому рекомендуется фрагменты которые редко выполняемые или требуют больших аппаратных затрат, реализовывать программно. Например, операции арифметики с плавающей запятой

рекомендуется реализовывать программно, а управление периферией – аппаратно;

2) Для предполагаемой области применения создаются библиотеки алгоритмов. Объект в таких библиотеках представляет собой структуру содержащую задачу, несколько вариантов программной реализации, реализующих структур, представленных на HDL языках описания аппаратуры Каждый из вариантов сопровождается количественными характеристиками, такими как время исполнения, затраты памяти, используемые ресурсы микросхем программируемой логики;

3) Оптимальное распределение частей задачи по исполнителям руководствуясь определенной целевой функцией, ограничениями и характеристиками задачи. В виде ограничений выступают имеющиеся ресурсы, такие как память, свободные макроячейки FPGA и т.д. Задача поиска оптимума является дискретной оптимизационной задачей. В виду того что много времени, затрачивается на решение оптимизационной задачи «точными» методами, такими как метод ветвей и границ, используются приближенные эвристические методы, которые позволяют сократить перебор и решить задачу выбора исполнителей с приемлемой точностью при сравнительно небольших затратах;

4) Разработка соответствующего интерфейса между процессором общего назначения и специализированным модулем, равно как и между блоками, включаемыми в аппаратную часть системы. При этом следует обращать внимание на такие проблемы, как согласованность форматов данных, буферизация, взаимное оповещение и взаимное блокирование процессов.

Процедуры сопряженного проектирования требуют наличия соответствующей аппаратной базы для их реализации.

Решение задачи устранения ошибок проектирования осуществляется путем комплексной отладки аппаратного и программного обеспечения. Появление противоречий в трактовке причин и источников отклонений характеристик системы от запланированных или ожидаемых могут привести к перераспределению функций между аппаратным и программным обеспечением.

Удобнее, чтобы стыковка произошла как можно раньше, и отладка выполнялась средствами, одинаково эффективными и для аппаратуры, и для программного обеспечения. Не менее важно, чтобы эти средства были одинаково удобны для разработчиков программного и аппаратного обеспечения. Вопросы разбиения или перераспределения задач между программным и аппаратным обеспечением могут возникать даже на этапе встраивания проекта в ИС или на этапах отладки готового изделия. Однако, чем позже, тем сложнее и дороже дается такое перераспределение.

Методология параллельного проектирования

В виде основных частей методологии параллельного проектирования выделяются:

1) формирование показателей качества объекта, которые используются для сбора и анализа требований;

2) определение методов выбора концепций (действенных подходов) для более эффективного построения объекта;

3) определение аксиом проектирования, таких как минимизация числа требований и ограничений на вычисляемые параметры; удовлетворение требованиям в порядке их важности;

4) определение правил проектирования, являющиеся систематизированными правилами положительного практического опыта и делящиеся на два класса. Для всех типов проектов применяется первое правило заключающееся в минимизации компонентов проектируемого объекта, использование симметричных компонентов и т.д. Второе правило направлено на синтез правил, применяемых в конкретной проблемной области [134].

Данная методология предполагает наличие распределенной управляемой среды разработки проектов, которая представляет собой совокупность объединенных в сеть автоматизированных рабочих мест (АРМ), позволяющих осуществлять доступ к общей базе проектов. В виду наличия общей информационной базы, повышенные требования предъявляются к защите

проектных решений, а также к информационному обмену между проектирующими подсистемами САПР [138].

Работа с каждым проектом осуществляется по принципу последовательно-параллельного конвейера, который позволяет обрабатывать проектируемые модули, образующие маршрут проектирования и реализующие проектные процедуры, а обрабатываемыми объектами являются фрагменты (узлы, блоки, детали), которые были получены в результате декомпозиции объекта проектирования. С переходом в такой режим работ, при котором формируется рабочую среду состоящая из множества конвейеров проектирования [145].

В случае проектных процедур обладающих различной производительностью, организуется асинхронный режим в конвейере проектирования. В подобной системе организуется централизованное хранилище проектных решений.

Методология сервис-ориентированной архитектуры

Методология SOA (Service Oriented Architecture), разрабатываемой корпорацией IBM [80], предполагает созданию современных корпоративных информационных систем с применением сервис-ориентированных архитектур.

SOA описывает набор бизнес-методов, методов процесса, организационных методов, методов управления и технических методов. Сервис-ориентированная архитектура предлагает гибкую работу с компонентами путем многократного использования и комбинирования. В такой архитектуре выделяются “информационная услуга” и “композитное приложение”.

Информационная услуга (сервис) представляет собой атомарную прикладную функцию автоматизированной системы, которая используется при разработке приложений.

Предлагаемый сервис характеризуется следующими свойствами:

1) многократность применения;

2) описание услуга одним или несколькими технологически независимыми интерфейсами;

3) слабая связанность услуг между собой, каждая из которых вызывается вызвана с помощью коммуникационных протоколов.

Композитное (составное) приложение представляет собой программное решение в определенной области, которое связывает логику процесса с источниками данных и информационных услуг. Композитные приложения ассоциированы с процессами деятельности и объединяют различные этапы, представляя их пользователю через единый интерфейс.

Данный подход при построении архитектуры сложных информационных интегрированных систем позволяет:

1) применять разные транспортные протоколы и стандарты форматирования сообщений;

2) обеспечить безопасность и надежность своевременной доставки сообщений;

3) повысить скорость разработки приложений и сократить затраты.

SOA имеет отношение к распределенным вычислениям и интеграции: от объектно-ориентированной парадигмы до концепций CORBA и DCOM. Концепции SOA используются в облачных технологиях.

Опишем возможности и преимущества применения методологий проектирования [108, 128, 134]. Результат сведем в табл. 1.4.

Таблица 1.4. Возможности и преимущества применения методологий проектирования

Показатель	**Методология IDEF3 Workflow**	**Методология сопряженного проектирования**	**Методология параллельного проектирования Concurrent Engineering (CE-проектирование)**	**Методология сервисно-ориентированной архитектуры (СОА)**
Поддержка коллективной работы и распределенных процессов	Путем координированного взаимодействия набором распределен	Путем оптимального распределение частей задачи по	Путем формирования множества конвейеров проектирования, составляющих	Путем обмена данными между независимо

	ных ресурсов	исполнителям	рабочую среду	действующими участниками
Декомпозиция	Проектные работы	Части проекта	Нет	Программные компоненты
Распределение задач между исполнителями	Да	Да	Да	Нет
Параллельная работа над несколькими решениями	Да	Да	Да	Да
Многократное использование	Да	Да	Нет	Да
Эффективность процесса управления	Оперативность, быстрота и компетентность, постоянный доступ к информации	Сокращение временных затрат и уменьшение числа итераций	Экономия времени (время от идеи до рынка сокращается на 25-50%), средства за счет повышения качества изделий, сокращение изменений (в 2-3 раза), вносимых в конструкцию на стадии изготовления	Тесное внутренние взаимодействие сервисов, улучшение процесса согласования работ подразделений, повышение гибкости организации, сокращение нагрузки на IT-подразделения
Организация работы	Наглядное представление контекста каждой задачи, позволяет проявлять гибкость в работе, быстроту исполнения	Проведение анализа и декомпозиция задач на фрагменты, которые будут реализованы аппаратно и/или программно	Использование проектных данных, с ранних стадий проектирования, одновременно различными группами специалистов	Использование распределенных, слабо связанных заменяемых компонентов, оснащенных стандартизированными

				интерфейсами для взаимодействия по стандартизированным протоколам
Контроль производительности выполнения задач	Повышение конфиденциальности и контроля доступа	Нет	Повышение требования к обеспечению защиты от несанкционированного доступа к проектным данным, а также к соблюдению регламентированных информационных обменов между проектирующими подсистемами	Удовлетворение политике и требованиям к качеству сервиса. Обеспечение масштабируемости, надежности, соблюдения соглашения об уровне обслуживания
Упрощение работы с данными	Удаленное хранение данных и использование их посредством Интернета	Создание библиотек алгоритмов	Распределенная компьютерная архитектура обеспечивает синхронизацию, оптимальное планирование и обработку информации	Ориентированность на бизнес, архитектурный подход, обеспечивающий интеграцию и повторное использование процессов и сервисов
Документирование и слежение за точным исполнением	Осуществляется, как запланировано	Нет	Руководитель проекта может осуществлять мониторинг и	Зависит от функционала сервиса

	руководством с учетом всех требований		контроль состояния проекта	
Управление процессом выполнения работ	Вмешательство только в случае исправления сбоев или последствий неправильного управления работой	При появлении противоречий осуществляется перераспределению функций между аппаратным и программным обеспечением	Вмешательство только в случае проведения организационных и технических мероприятий с целью изменения технологического процесса или модификации организационной структуры	Формирование блочной системы, в которой можно собирать из блоков-сервисов требуемое комплексное ИТ-решение, «склеивая» их между собой. Размер сервиса не должен сдерживать изменения процесса и должен быть таким, чтобы им можно было свободно оперировать на уровне изменяемых бизнес-процессов.

Рассмотрение этих методологий в совокупности позволяют описать процесс распределенного проектирования и спроектировать архитектуру системы обмена информацией между независимо действующими участниками.

1.5. Многоагентные системы проектирования

Применение интеллектуальных электронных систем – так называемых «умных» сред (smart environments) является наиболее известным направлением

технологического развития. Одной из области применения данной технологии является конструкторско-технологическое проектирование.

Современный этап автоматизации машиностроительных предприятий в ведущих странах мира характеризуется интенсивной разработкой и широким внедрением CAD/CAM/CAE/PDM и CALS систем, включающих интеллектуальные компоненты [157].

В качестве основной идеи программных агентов выделяется делегирование полномочий, т.е. агенты выполняют свою работу в фоновом режиме от лица пользователя при решении различных задач. Взаимодействие агентов с пользователем необходимо для получения заданий и установок. Также агент должен возвращать полученные результаты, ориентироваться во внешней среде и принимать решения, для выполнения задачи. Многоагентная система (МАС) является программно-вычислительным комплексом, где взаимодействуют различные агенты (общаются, кооперируются и договариваются между собой) для решения поставленных перед ними задач.

МАС описывается в виде совокупности взаимосвязанных агентов, способных взаимодействовать друг с другом и окружающей средой, обладающих определенными интеллектуальными способностями и возможностью индивидуальных и совместных действий.

При построении многоагентных систем можно выделить два класса систем:

1) распределенные МАС с “высокоинтеллектуальными” агентами;

2) МАС основанные на “групповом разуме” (swarm intelligence).

В первом классе, внимание уделяется распределенным системам, в которых на отдельном сервере находятся небольшое количество агентов (единицы, десятки), и каждый из агентов обладает сложной логикой функционирования и возможностью межмашинного обмена данными.

Во втором классе внимание уделяется одному серверу, содержащему большое количество агентов [7,11] (тысячи, десятки тысяч). В данной системе агент не является сложным объектом и реализует простые алгоритмы

переговоров. Путем взаимодействия нескольких агентов решаются сложные задачи.

Общая архитектура многоагентной системы представлена на рис. 1.11.

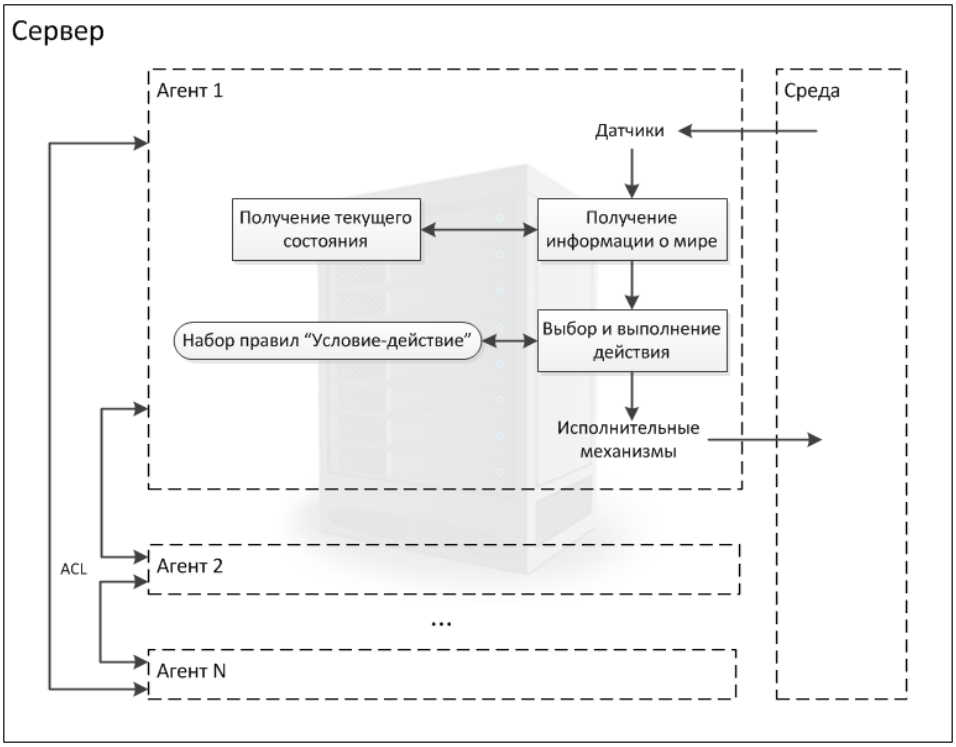


Рис. 1.11. Архитектура многоагентной системы

Методологии проектирования многоагентных систем

Методологии агентно-ориентированного анализа применяются на стадиях анализа и проектирования многоагентных систем. Выделяют четыре класса методологий: базирующиеся на объектно-ориентированных методах и технологиях с использованием соответствующих расширений (Agent UML, P2P Agent Platform), использующие традиционные методы инженерии знаний (MAS-Common KADS), основанные на организационно-ориентированных представлениях (Gaia, ПВ-сети, М-архитектура), комбинирующие в различной степени методы трех первых классов (Tropos, PASSI, Prometeus).

Инструментальные среды используются на стадиях реализации и тестирования многоагентных систем. Они используют свои модели, соответствующие этому уровню детализации системы. Можно выделить два основных класса инструментальных сред: фреймворки (JADE) и среды разработки (Zeus, Agent Builder).

Методология для агентно-ориентированного анализа и проектирования Gaia представляет собой общий метод - пригодный к широкому ряду многоагентных систем, а также может использоваться как на макроуровне (общества), так и на микроуровне (агент) системы. Gaia базируется на том, что многоагентная система – вычислительная организация, основанная на взаимодействии различных ролей. Основные этапы Gaia-методологии представлены на рис. 1.12.

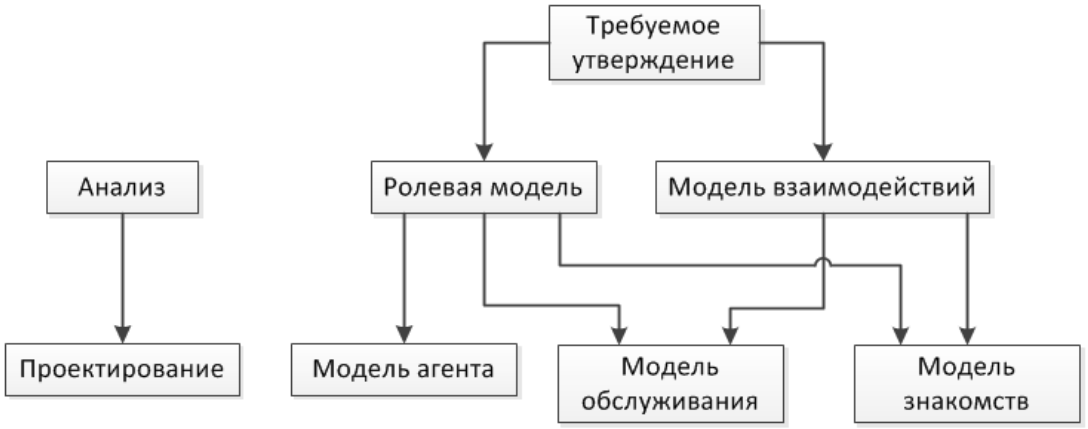


Рис. 1.12. Основные этапы Gaia-методологии

САПР, использующие многоагентные системы

Существующие варианты применения многоагентных систем в САПР используются для создания систем поддержки коллективной работы проектировщиков или распределения решения задачи между агентами [105,107,131,133].

Помимо основных целей МАС обеспечивает сокращение трудоемкости и стоимости сопровождения и развития систем за счет использования автоматической генерации программных средств, а также модульности, открытости и простоты модернизации систем проектирования.

Многоагентные системы на основе сценариев ориентированы на использование в больших компьютерных сетях. Агенты данного класса систем разрабатываются с помощью интерпретируемых языков Java, PHP и др. Использование интерпретируемых языков облегчает поддержку гетерогенных систем. Эти языки ориентированы на реализацию асинхронного процесса и удаленное исполнение приложений.

Для реализации МАС, основанных как на статических, так и на динамических распределенных приложениях, наиболее перспективными являются следующие технологии: DCOM (Microsoft Distributed Component Object Model), Java RMI (Java Remote Method Invocation) и CORBA (Common Object Request Broker Architecture) [132].

Взаимодействие пользователя с распределенными компонентами САПР существенно упрощается при использовании интернет/интранет решений для разработки пользовательского интерфейса. В этом случае компоненты САПР (клиенты) могут быть интегрированы посредством так называемой «шины представительского уровня».

Использование распределенности в структуре многоагентной системы может быть представлена в виде архитектуры представленной на рис. 1.13.

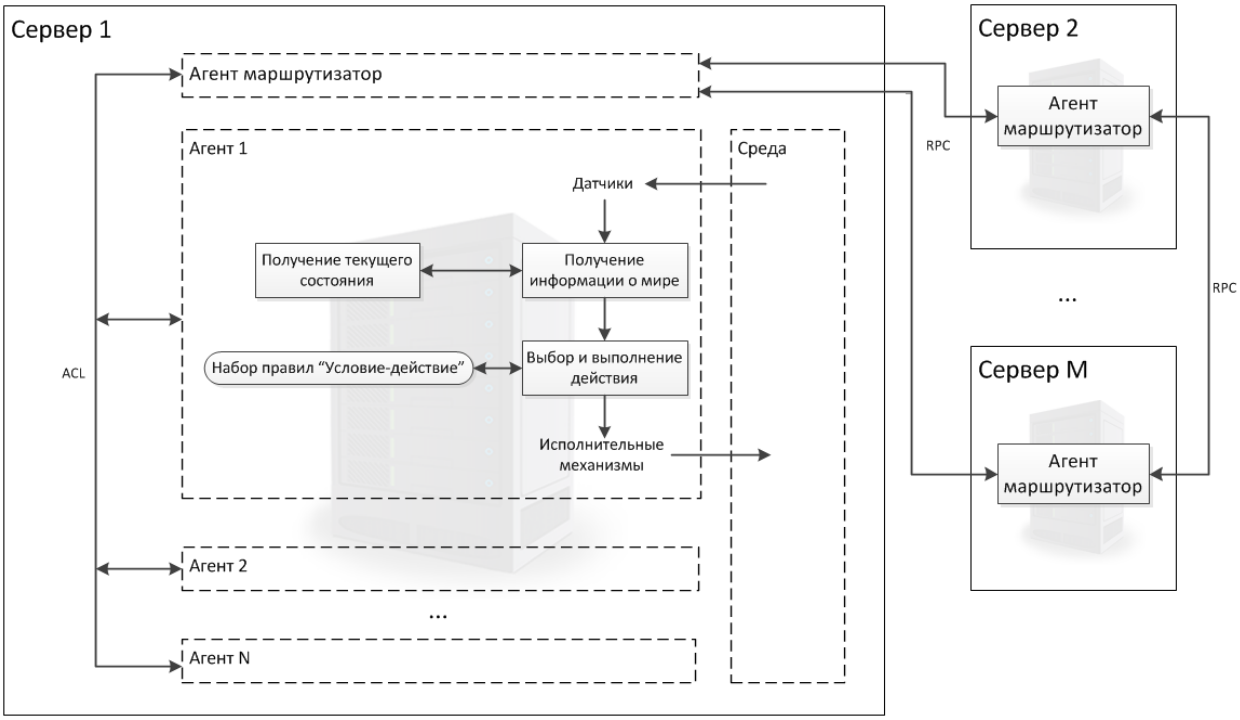


Рис. 1.13. Архитектура многоагентной СРП

Системы использующие многоагентную технологию для автоматизации проектирования [105, 107] в основном направлены на решение задач связанных с процессами расчета, поиска данных, оформлению результатов, решения сложных задач управления ресурсами и управления производством изделий, от поступления заказа до выпуска продукции. Агенты обладают возможностью

синтеза структуры представляемых ими объектов и подбора параметров элементов их составляющих. Также агент может получить информацию о характеристиках объектов в виде макромоделей. В некоторых системах есть поддержка работы групп разработчиков, которые могут находиться на удаленном расстоянии друг от друга.

В виду этого, актуальным является применение многоагентной технологии при работе с СФЛМ, использующими языки описания, такие как VHDL. Инструментальные средства должны обеспечивать генерацию текстового представления из базы данных, на языке который будет понятен проектировщику, и генерацию эффективных программных кодов, для улучшения которых не нужно привлекать программистов. Новый подход должен быть ориентирован на иной способ использования интеллектуальной многоагентной системы для работы с СФЛМ. Решение исходной задачи проектирования должно быть получено в результате процессов кооперации и конкуренции внутри многоагентной системы, т.е. роль проектировщика, в основном, сводится к постановке задачи и формированию программного кода, если необходимое решение не было найдено в базе данных.

Представим в виде табл. 1.5 применение многоагентных систем с описанием их типа и возможностей [80, 8, 23, 104, 116, 131].

Таблица 1.5. Применение многоагентной системы

Показатель	**WebCADby Features WebCAPP WebTurning**	**DESO**	**САПР ВАЛ**	**LOD**	**INFELT STEP**
Тип системы	Автоматизированная система технологической подготовки производства	Многокомпонентная распределенная САПР на основе концепции параллельного инжиниринга	САПР технологического процесса	САПР 3D проектирования [8]	Автоматизированная система планирования процессов
Поддержка	Да	Да	Нет	Да	Да

параллельного проектирования					
Возможность функционального расширения агентов пользователями	Нет	Да	Да	Нет	Нет
Средства разработки архитектуры и взаимодействия между удаленными объектами	Virtual Reality Markup Language (VRML), Java Agent Template Lite (JATLite)	CORBA, DCOM, ActiveX и Java	JADE	JADE	Платформа разработана с помощью Visual Studio 2010 и SQL Server 2008. Агенты были созданы на основе объектно-ориентированной концепций.
Задачи решаемые системой	Дистанционное изготовление цилиндрических деталей через Интернет [4]	Среда проектирования структурированных объектов предназначена для проектирования гибких производственных систем [131,133]	Технологический процесс ковки ступенчатых валов [116]	Совместное 3D-проектирование, используя модель продукта на различных уровнях детализации LOD [108]	Проведение тематического исследования. Отображение геометрических данных формата. DXF в виде стандартной структуры STEP и наоборот [3,20,23]

Концепция языка агентов для поддержки моделей взаимодействия	KQML	KQML	FIPA-ACL	FIPA-ACL	KQML

Несмотря на то, что внедрение технологии многоагентных систем является довольно сложной задачей, требующей точного понимания предметной области, основных понятий и концепций архитектуры МАС и наличия представления о современных концепциях проектирования подобных систем, применение многоагентной технологии в САПР является перспективным подходом.

1.6. Постановка задачи

Проведенный выше анализ применяемых методологий, методов и средств распределенного проектирования, обработки и представления описания проектируемых устройств на языке описания аппаратуры показывает.

1. Существующие программные средства получения проектных решений только частично используют современные web-технологии с целью организации распределенного проектирования.

2. Отсутствие свойства кроссплатформенности у систем типа ECAD ограничивает возможности использования при удаленной работе в различных операционных системах.

3. Отсутствие прямого взаимодействия процесса управления проектами и задачами, а также их решениями в одной системе приводит к разделению внимания проектировщиков на несколько параллельно функционирующих систем при наличии подобных, т.к. являясь отдельными системами, они могут отсутствовать у компании в виду определенных затрат для ее покупки, установки и настройки.

4. Остается актуальным формирование библиотеки проектных решений с целью их повторного использования для получения нового решения и формирования “вычислительных заготовок”.

5. Существующие системы проектирования типа ECAD обладают хорошим функционалом для разработки устройств и моделирования их работы, но являются закрытыми системами, и поэтому интеграция пользовательских решений в виде плагинов остается затруднительной в виду отсутствия или ограничения на использование API системы.

6. При наличии в существующих ECAD системах возможности наполнения проектировщиком библиотек элементов и шаблонов описаний HDL структур, оставлены без внимания методы формирования шаблонов структурно-функционального представления из HDL-описания проектируемого устройства, которое описывается в единой форме для удобства последующего использования.

7. В существующих ECAD системах отсутствуют интеллектуальная составляющая, связанная с процессом формирования проектных решений, в виде совокупности интерфейсов, описаний функционирования и структур проектируемых устройств.

На основании вышесказанного и проведенного в первой главе данной работы анализа предметной области, сформулируем основные исследовательские задачи.

1. Анализ методологий, методов и средств построения многоагентных СРП и организации коллективного проектирования.

2. Проектирование многоагентной системы для работы со структурно-функциональными лингвистическими моделями на языке описания аппаратуры VHDL в процессе коллективной разработки.

3. Разработка концептуальной модели системы распределенного проектирования (СРП) СФЛМ на базе цветных сетей Петри и проведение анализа на ее основе с целью верификации СРП, проверки сценарного взаимодействия агентов СРП.

4. Разработка методов управления проектными задачами и процессом получения проектных решений.

5. Разработка методов формирования, синтеза и поиска СФЛМ в СРП.

6. Разработка программного обеспечения, позволяющего проектировщикам создавать сложные устройства, оперируя объектами СФЛМ, находясь на удаленном расстоянии друг от друга.

1.7. Выводы

1. Коллективное проектирование позволяет эффективно распределять работу над проектом между несколькими проектировщиками, которые могут находиться, как в одном месте, так и на удаленном расстоянии, при этом сохраняя контроль над проектом и системой.

2. Обеспечение актуальности данных проекта и автоматической синхронизации изменений, являясь одними из основных факторов успешного коллективного проектирования, позволяют организовать взаимодействие повышающее качество и скорость получения проектного решения.

3. Большинство современных САПР электронных объектов являются монолитными и обладают двумя основными негативными факторами: ограниченной гибкостью при внесении определенных изменений в функционал системы и высокой стоимостью как системы, так последующего внесения изменений. В виду высокой конкуренции и постоянных изменениях к требованиям повышения качества и функциональности проектируемых устройств, эти факторы являются критичными для деятельности современных компаний.

4. Многоагентный подход является одной из самых перспективных технологий проектирования архитектуры сложных систем за счет синтеза различных традиционных технологий при построении и функционировании агентов. Такой подход обладает гибкостью, устойчивостью и масштабируемостью при изменении границ решаемых задач.

5. Применение многоагентной технологии в САПР позволяет повысить интеллектуальность процесса от формирования проектных задач и их распределения между проектировщиками, до получения полноценного проектного решения, а также улучшить эффективность и адаптируемость самой системы.

6. Анализ существующих методов получения проектных решений показывает, что наиболее эффективным является формирование проектируемого модуля в виде вычислительной заготовки, которая хорошо описана, имеет понятный интерфейс, документацию, комментарии и испытательный стенд с надежными тестами. Все эти атрибуты позволяют повторно применять описываемый модуль в любом новом проекте.

7. Рассмотренные системы получения проектных решений не предоставляют возможности интеграции со сторонними сервисами в целях получения проектных решений в виде HDL описаний или вычислительных заготовок, а, следовательно, проектировщику приходится либо ограничиваться библиотеками самой системы, либо искать нужное решение в сети Internet, либо проектировать устройство с нуля.

8. Высокая сложность современных СБИС и систем на их основе требует систематической методологии проектирования, сопровождаемой четкой спецификацией, позволяющей формировать задачи на разработку устройства и распределять их между проектировщиками.

9. Одним из важных факторов, гарантирующих сохранность разработанных или разрабатываемых проектных решений в ходе проектирования, является бэкапирование, которое отсутствует в большинстве рассмотренных систем или реализовано в виде системы контроля версий.

10. Разработка коллективной среды распределенного проектирования на основе многоагентного подхода позволит систематизировать, регламентировать взаимодействие членов команды в процессе проектирования и сформировать у них компетенции в области коллективного проектирования вычислительных устройств, что, в конечном итоге, обеспечит сокращение сроков и повышение качества создания проектных решений.

Указанные недостатки нашли отражение в сформулированных задачах исследования с целью их устранения при проектировании СРП.

ГЛАВА 2. РАЗРАБОТКА МНОГОАГЕНТНОЙ СТРУКТУРЫ СИСЕМЫ РАСПРЕДЕЛЕННОГО ПРОЕКТИРОВАНИЯ СФЛМ

В данной главе предлагается структура многоагентной системы распределенного проектирования, описываются ее подсистемы. Приводятся типы, описание функционального назначения агентов и сценарии их взаимодействия.

Разрабатывается процесс получения проектных решений от описания на VHDL языке до формирования СФЛМ. Разрабатывается модель в виде параллельной сетевой схемы задач (ПССЗ), предназначенной для управления процессом распределенного проектирования. Рассматриваются типы связей между задачами и операторами, а также шаблоны формирования операторов в ПССЗ.

2.1. Организация многоагентной системы распределенного проектирования

Обобщенная структура СРП СФЛМ представлена на рис. 2.1. В ее основе лежит технология CORBA, позволяющая организовать единую информационную среду, компоненты которой могут взаимодействовать друг с другом вне зависимости от их конкретной реализации [122,130,137]. На основе формирования промежуточного программного обеспечения (ППО, middleware) [141] производится связывание программных приложений, которые находятся на разных серверах [54,55,56]. Идея создания корпоративной САПР рассматривалась в [146].

В рассматриваемой архитектуре используется частное облако как средство организации процесса распределенного проектирования с централизованным хранением проектных решений и доступом как в локальной сети предприятия, так и через сеть Internet. Под частным облаком будем понимать инфраструктуру, предназначенную для использования одной организацией, включающей несколько подразделений. Также в описываемой архитектуре для частного облака описывается возможность применения бизнес-модели SaaS (Software as a Service), позволяющую использовать программное обеспечение как услугу.

Слабым местом бизнес-модели SaaS являются вопросы безопасности и возможной утечки информации со стороны поставщика этих услуг [69]. Вопросы безопасности данных ограничивают использование SaaS для систем, в которых обрабатывается конфиденциальная информация, например, продукты интеллектуальной собственности. В рассматриваемой СРП таким продуктом является проектное решение СФЛМ [73]. Обеспечение безопасности и конфиденциальности связано с созданием частного облака, которое вместе с локальной сетью является аппаратно-технологической основой корпоративной СРП. Соответствующее программное обеспечение устанавливается на серверах предприятия с целью получения доступа только для сотрудников данного предприятия и сохранения разработанных проектных решений на локальном сервере [114]. Организация связи с облачным сервисом необходима для получения данных с удаленных систем. Такое взаимодействие удаленных систем позволяет построить корпоративную сеть обмена проектными решениями. Для организации взаимодействия между серверами предприятий и облачным сервисом предлагается использовать технологию CORBA [152].

В качестве функциональных модулей СРП выделены следующие подсистемы: управления проектными задачами, маршрутизации, разработки проектного решения, поиска шаблона СФЛМ, тестирования и оптимизации [69]. На рис. 2.1. показаны внутренние связи между подсистемами и внешние связи между серверами с указанием протокола взаимодействия.

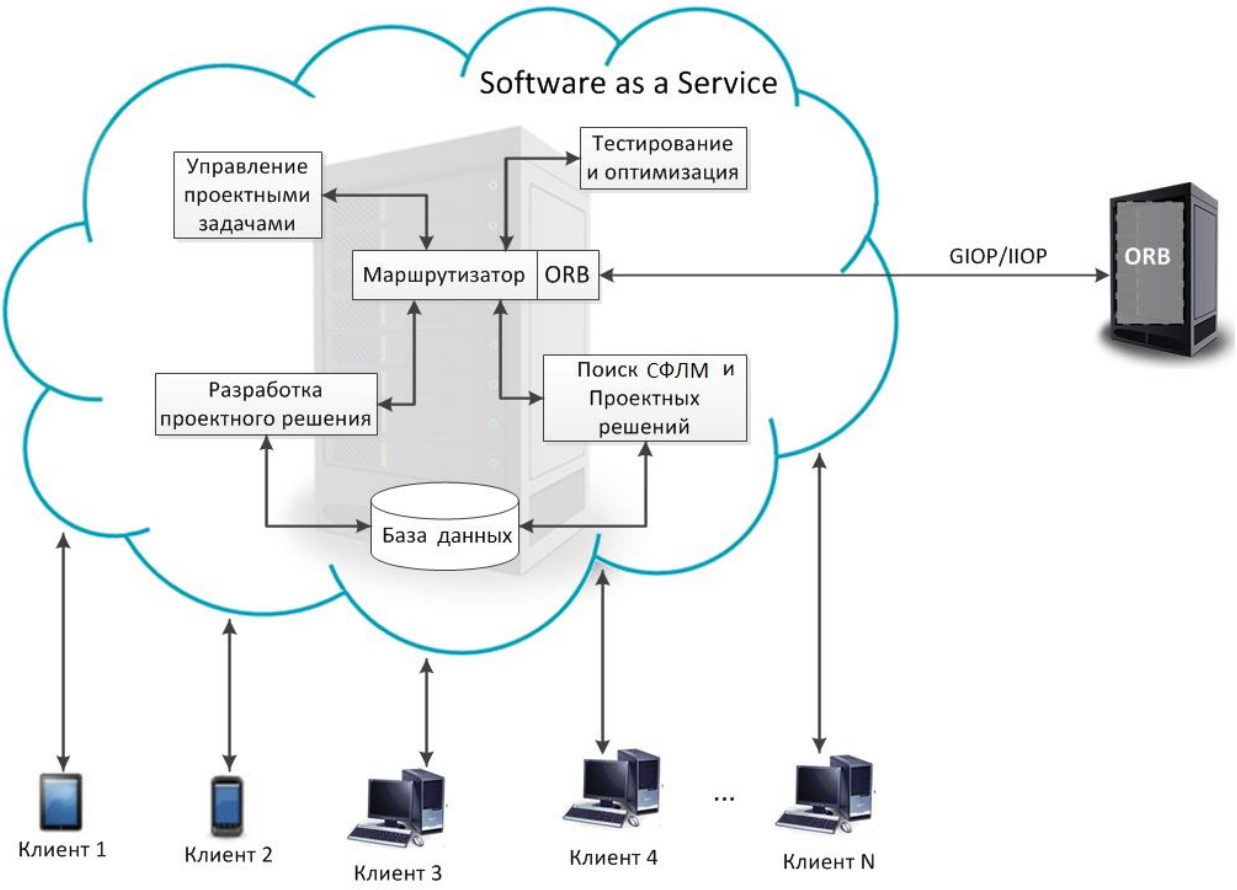


Рис. 2.1. Структура системы распределенного проектирования СФЛМ

Подсистема управления проектными задачами

Основным назначением данной подсистемы является распределение выполнения проектных задач между проектировщиками с целью оптимизации использования ресурсов и сокращения времени создания проектного решения. На рис. 2.2 представлены ее основные функциональные блоки. Целью работы балансировщика является распределение процесса выполнения проектных задач между несколькими проектировщиками для оптимизации использования ресурсов и сокращения времени разработки проектного решения [70].

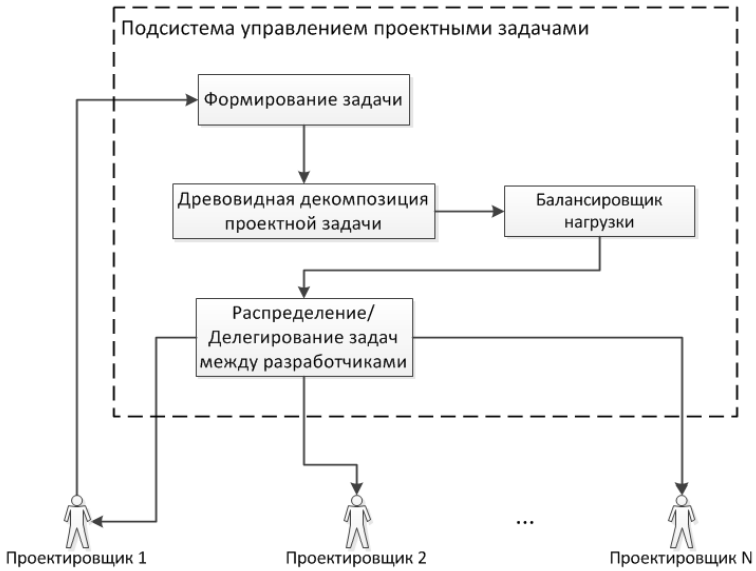


Рис. 2.2. Подсистема «Управление проектными задачами»

Подсистема маршрутизации

Подсистема предназначена для организации взаимодействия между элементами системы. Ее основным функциональным назначением является передача проектных задач, проектных решений и формирование маршрута проектирования. Частью подсистемы является ORB, который отвечает за поиск реализации объекта, его подготовку к получению и обработке запроса, передачу запроса и доставку результатов клиенту, а также совокупность сервисов, интерфейс и API для обращения к ORB.

Объектный адаптер предоставляет все низкоуровневые средства для связи объекта с его клиентами: генерацию ссылок на удаленные объекты; вызов методов, определенных в IDL; обеспечение безопасности взаимодействия; активацию и деактивацию объектов; установление соответствия между ссылками на удаленные объекты и реальными экземплярами объектов; регистрацию объектов.

В подсистеме маршрутизации ведется логирование событий системы. Это позволяет фиксировать ошибки в процедурах формирования проектных решений, синтеза, оптимизации и тестирования.

Подсистема разработки проектного решения

На основе результатов анализа объекта моделирования проектировщик строит структуру и алгоритм функционирования модели. В зависимости от размеров схемы структура может быть описана на языке описания аппаратуры VHDL или в графическом виде, где каждому будущему объекту из БД ставится в соответствие черный ящик с входами/выходами. Описание на языке система строит самостоятельно. Для выбранного шаблона СФЛМ рассчитываются входные параметры. Процесс верификации заключается в установлении соответствия проектного решения и спецификации. После создания нового проектного решения данные сохраняются в базе.

Модуль синтеза проектных решений позволяет сформировать новое проектное решение на основе данных, которые получены от проектировщиков. Модуль интеграции позволяет объединить описание структуры на языке VHDL с описанием структуры в графическом виде и создать полноценное проектное решение. На рис. 2.3 представлены основные функциональные блоки подсистемы разработки проектного решения и процесс их взаимодействия.

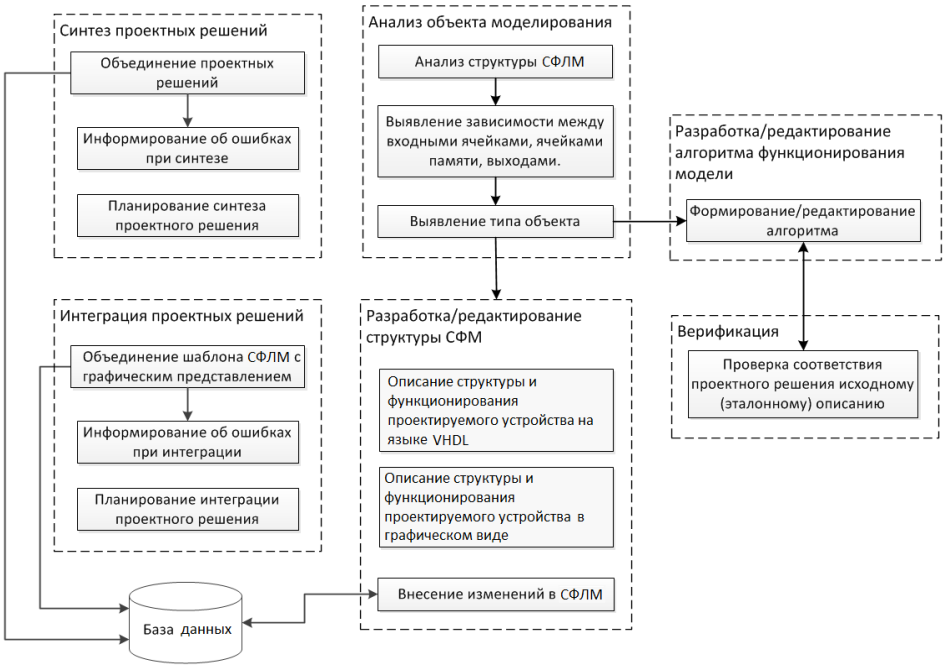


Рис. 2.3. Подсистема «Разработка проектного решения»

Подсистема поиска шаблона СФМ

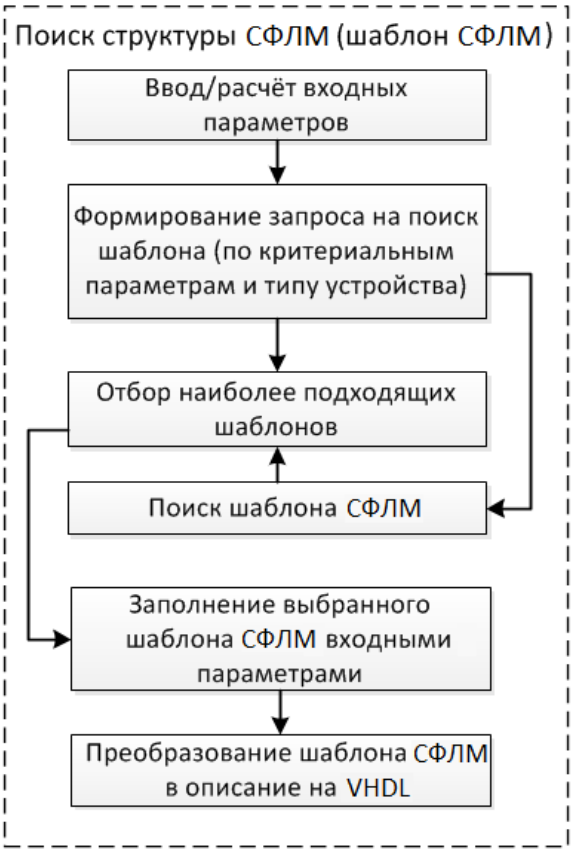


Рис. 2.4. Подсистема «Поиск шаблона СФМ»

Подсистема поиска необходима для использования шаблонов, которые были сохранены в базе данных системы проектирования.

Поиск шаблонов СФЛМ ведется по критериальным параметрам и типам устройств. Из всех шаблонов выбирается наиболее подходящее описание СФЛМ, которое преобразуется в описание на языке VHDL [62].

В структуре шаблона присутствует набор входных параметров. Расчет параметров производится с помощью алгоритма функционирования модели.

На рис. 2.4 представлены основные функциональные блоки подсистемы поиска и процесс их взаимодействия.

Подсистема тестирования и оптимизации

Подсистема позволяет оценить окончательную программную модель и провести ее финальное тестирование. При выявлении несоответствий разработанной модели и требований, вносятся изменения в параметры. Если изменение параметров не приводит к положительному результату, данные передаются в подсистему разработки проектного решения, в котором вносятся изменения в структуру СФЛМ. Подсистема позволяет исследовать разработанную модель. Целью исследований является определение выходных характеристик модели при разных значениях управляемых переменных параметров модели. Выбор входных параметров является необходимым этапом

в эксперименте. На этом этапе необходимо определить наиболее значимые параметры, т.к. в моделях средней и большой сложности невозможен полный перебор всех входных параметров. В ходе исследования проводятся измерения критериальных параметров, и производится их постепенное улучшение [62]. В процессе проектирования может быть протестировано несколько вариантов, используя подсистемы поиска и разработки проектного решения. Это позволит значительно сократить время на разработку нового устройства. На рис. 2.5 представлены основные функциональные блоки подсистемы тестирования и оптимизации, а также процесс их взаимодействия.

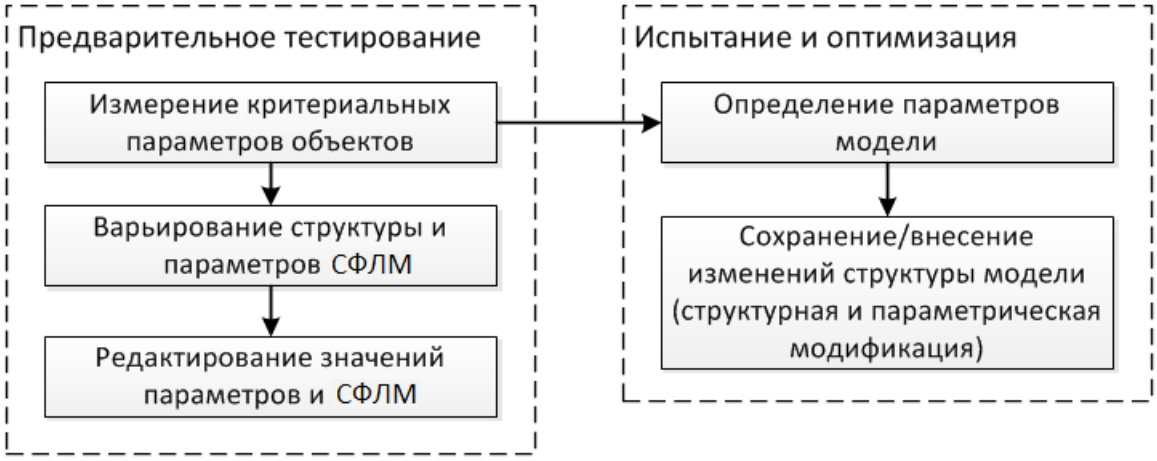


Рис. 2.5. Подсистема «Тестирование и оптимизация»

2.2. Типы и функции агентов

В диссертационной работе для выполнения и управления потоками работ предлагается использовать многоагентный подход, как наиболее подходящий в плане учета специфики содержания указанных потоков. В качестве гипотез позитивов от применения многоагентного подхода является сокращение времени проектирования и повышение качества формирования проектных решений, которое доказывается в ходе диссертационного исследования.

Типовая схема, составленная на основе анализа [136,96,118,97] проектирования ВУ, включая описание на HDL, представлена на рис. 2.6 и содержит следующие этапы и задачи.

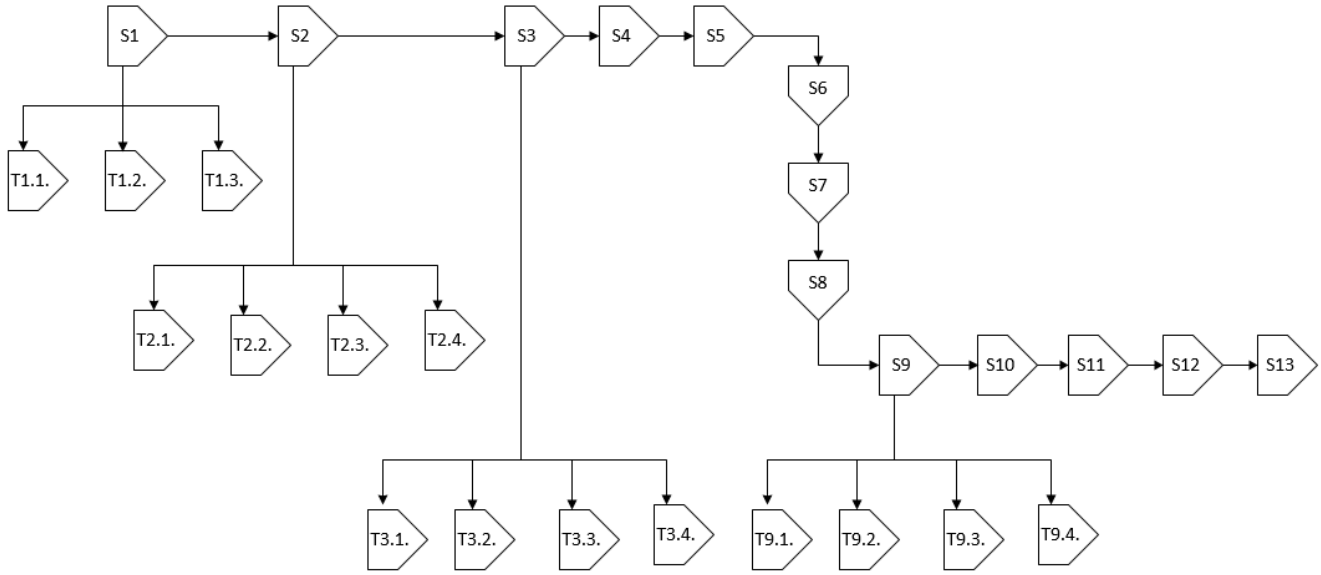


Рис. 2.6. Диаграмма потоков работ проектирования ВУ на HDL

Опишем этапы и задачи, которые были представлены на диаграмме.

1. Концепция. Выработка и формализация идеи

1.1. Формирование технического задания

1.2. Выбор соответствующего стиля реализации проекта (design style), который, должен быть наиболее пригодным к реализации проекта и его последующей верификации

1.3. Определение соответствующих шагов в маршруте проектирования, а также используемых библиотек

2. Разработка иерархической блок-схемы проекта. Определение базовых функций составных частей СБИС. Формирование проекта архитектуры.

2.1. Описание структуры системы

2.2. Описание декомпозиции системы на подсистемы

2.3. Описание спецификации связей

2.4. Описание взаимодействия подсистем

3. Разработка технических требований (specification). Ввод проекта в форме схемных решений блоков, которые реализуют выбранную архитектуру.

3.1. Спецификации функционирования системы

3.2. Спецификации функционирования узлов

3.3. Спецификации функционирования блоков

3.4. Спецификации реализуемых функций

4. Прототипирование аппаратно-программных частей проекта

5. Описание проекта с помощью одного из языков описания аппаратуры, как правило VHDL или Verilog, на уровне регистровых передач (register transfer level, RTL)

6. Функциональная верификация модели. Моделирование системы и ее работы на основе четкой спецификации структуры системы, а также функционирования ее компонентов

7. Процесс компиляции и оптимизации (синтез)

8. Формирование списка связей проекта и выполнение «тонкой настройки»

9. Трассировка

9.1. проверка и трансляция списка связей

9.2. размещение элементов проекта в виртуальных конфигурируемых логических блоках

9.3. замена виртуальных конфигурируемых логических блоков реальными и трассировка каналов соединений

9.4. создание файла конфигурации (bit stream)

10. «Ручное» размещение логики ПЛИС

11. Формирование файлов-отчетов о результатах реализации проекта

12. Тестирование

13. Передача в производство и осуществление последующего тестирования образцов

Ниже приведены намерения по применению агентов для управления потоками работ.

Агентно-ориентированные технологии являются одним из активно развивающихся направлений информационных систем и применяются во многих предметных областях [58,74,86,155]. Многоагентная система представляет собой совокупность взаимосвязанных агентов, способных взаимодействовать друг с другом и окружающей средой, обладающих определенными интеллектуальными

способностями и возможностью индивидуальных и совместных действий [54,115, 142,143].

В [124] рассматриваются следующие типы агентов: рефлексный агент; рефлексный агент, основанный на модели; агент, основанный на цели; агент, основанный на полезности.

В СРП к рефлексным агентам относятся интерфейсный агент и агент базы данных; к рефлексным агентам, основанных на модели, - агент рабочей памяти; к агентам, основанных на полезности, - агент управления проектными задачами, поисковый агент и агент разработки проектного решения; к агентам, основанным на цели, - агент синтеза проектных решений и агент маршрутизации.

Интерфейсный агент (interface agent [INA]) выполняет связующую роль агентов.

Агент управления проектными задачами (management agent project tasks [MAPT]) выполняет формирование ПССЗ и распределение проектных задач между проектировщиками. Данный агент формирует процесс проектирования, работая непосредственно с ПССЗ. Формирует операторы, ищет существующие ПССЗ или операторы, отсортированные по релевантности, ищет исполнителей и контролирует ход выполнения задач путем информирования проектировщиков о проектных задачах, обновление ПССЗ с учетом появления новых задач и их приоритетов. Использование агента на этапах 2-3 позволит сократить время на формирование проекта и уменьшить влияние человеческого фактора.

Агент разработки проектного решения (agent designer project solution [ADPS]) выполняет операции, связанные с созданием проектного решения на языке VHDL, т.е. формированием структурно-функциональной лингвистической модели (СФЛМ), шаблона СФЛМ и проведением лексического и семантического анализа кода. Данный агент начинает свою работу на 5 этапе. Создавая СФЛМ из описания на VHDL, он проверяет соответствие разработанных моделей, спецификациям, которые были разработаны ранее. Формирует и представляет отчет о проведенном анализе и этим сокращает время на верификацию проектных решений. Также данный агент на этапе 12 работая в

полуавтоматическом режиме, формирует набор тестов и прогоняет их на созданном проектном решении. Формирует и представляет отчет о проведенном тестировании и этим сокращает время на тестирование проектных решений.

Агент синтеза проектных решений (agent synthesis project solution [ASPS]) выполняет поиск готовых к объединению проектных решений, созданных проектировщиками, а также их синтез в единое проектное решение. Данный агент работает на 2, 3, 5 и 12 этапах. В зависимости от проектных этапов меняется характер синтеза. Синтез описаний и спецификаций отличается от синтеза vhdl кода тем, что они имеют структурированный вид, позволяющий обрабатывать данные в последующей работе над проектом. Взаимодействуя с агентом управления проектными задачами, данный агент, через систему контроля версий в полуавтоматическом режиме объединяет сформированные блоки проектируемого устройства. Этап формирования тестов, аналогичен формированию проектных решений.

Агент маршрутизации (router agent [ROA]) выполняет связующую роль между локальными или распределенными агентами, расположенными на других серверах, которые выполняют роль хранения или разработки проектных решений.

Поисковый агент (search agent [SEA]) выполняет формирование запроса на поиск проектного решения и кластеризацию данных с целью сокращения времени поиска данных. Данный агент работает на этапах с 1, 2, 3, 5. Он анализирует ситуацию, в которой находится проектировщик и информирует его о потенциальных решениях путем предоставления данных о проектных задачах и проектных решениях, которые были реализованы ранее.

Агент базы данных (agent data base [ADB]) выполняет операции по работе с базой данных.

Агент рабочей памяти (agent working memory [AWM]) управляет состоянием системы и распределением нагрузки как на агентов, так и на всю систему.

Структура многоагентной СРП представлена на рис. 2.7.

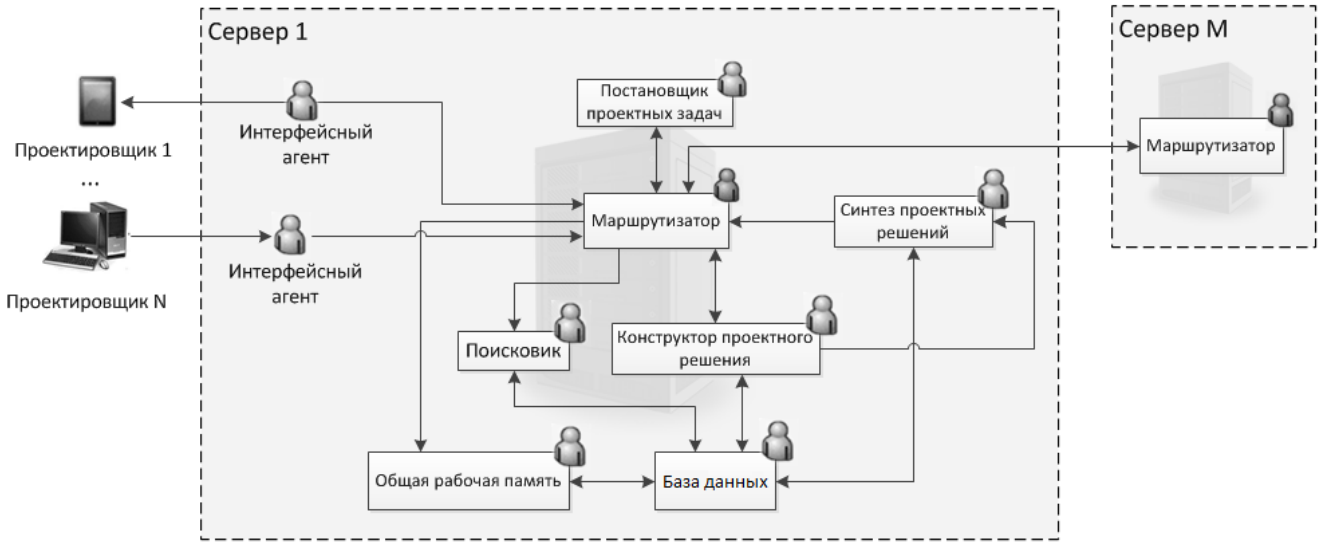


Рис. 2.7. Структура многоагентной системы распределенного проектирования

Проведение синтаксического и семантического анализов кода VHDL-программы и формирование СФЛМ [63] на основе данного кода позволяет агенту разработки проектного решения формировать библиотеку проектных решений, используя которую поисковый агент осуществляет поиск решения для разрабатываемого компонента с учетом его спецификации. Основные операции, выполняемые агентом синтеза проектных решений, обеспечивают получение проектных решений путем объединения СФЛМ отдельных компонентов разрабатываемого устройства.

2.3. Сценарии взаимодействия агентов

В зависимости от ситуации, возникающей при работе в СРП, в сообщениях агентов содержится различная информация, поэтому для анализа процесса выделим следующие типы взаимодействий.

1. Receive Approval (RA) – позволяет агенту принимать сообщения от другого агента.

2. Accept Requests (AR) – позволяет агенту принимать запросы от другого агента.

3. Send approval (SA) – позволяет агенту посылать сообщения другому агенту.

Согласно спецификации FIPA (Foundation for Intelligent Physical Agents) выделим следующие типы протоколов взаимодействия.

1. Request Interaction Protocol (IP) – протокол взаимодействия, позволяющий агенту запросить другого агента выполнить некоторое действие.

2. Query Interaction Protocol (QIP) – протокол взаимодействия, позволяющий одному агенту запросить выполнение некоторого действия на другом агенте.

3. Request When Interaction Protocol (RIP) – протокол взаимодействия, позволяющий агенту запросить выполнение некоторого действия, учитывая определенные условия.

Для наглядного представления взаимодействия агентов сформирован граф, вершинам которого соответствуют агенты, а дугам – обмен сообщениями в процессе коммуникации. Последовательность посылаемых агентами сообщений образует сценарий их взаимодействия. Например, при поиске проектного решения сценарий взаимодействия агентов может быть представлен в виде графа, изображенным на рис. 2.8. Т.к. сценарий представляет собой последовательность сообщений передаваемых агентами, то над дугами в графе указываются номер сообщения [72].

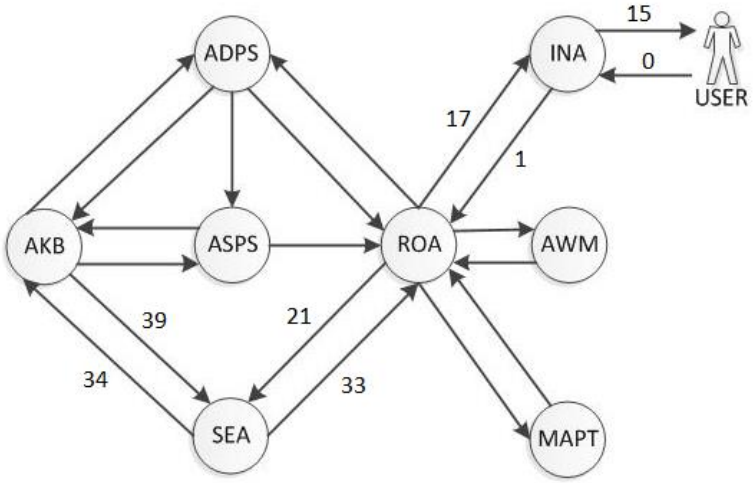


Рис. 2.8. Граф сценария взаимодействия агентов при поиске проектного решения

В табл. 2.1 представлены сценарии взаимодействия агентов, отражающие соответствия между типом сообщения, порядком его передачи и действиями агента.

Таблица 2.1. Сценарии взаимодействий агентов

Агент-отправитель (sender)	Агент-получатель (receiver)	Тип	№ сообщения	№ пред. сообщения	Описание взаимодействия
INA	ROA	IP	1	–	посылает запрос с данными для поиска проектного решения
		IP	2	–	посылает запрос с данными для формирования/редактирования СФЛМ
		IP	3	–	посылает запрос данными для формирования/редактирования шаблона из СФЛМ
		IP	4	–	посылает запрос с данными для поиска проектировщика, который может решить текущую проектную задачу
		IP	5	–	посылает запрос с данными для поиска проектной задачи в ПССЗ и совокупности связанных задач, следующих за найденной задачей
		IP	6	–	посылает запрос с данными для формирования оператора ПССЗ описывающего проектную задачу
		IP	7	–	посылает запрос с данные для внесения изменений в сформированный оператор ПССЗ
		IP	8	–	посылает запрос для удаления оператора ПССЗ с последующей перестройкой ПССЗ
		IP	9	–	посылает запрос анализа VHDL кода на наличие ошибок

		IP	10	–	посылает запрос с данными для формирования/редактирования VHDL кода
		IP	11	–	посылает запрос с данными поиска ПССЗ для нового проекта
		IP	12	–	посылает запрос на формирование графического представления СФЛМ
	USER	–	13	–	предоставление результатов регистрации
		–	14	–	предоставление результатов авторизации
		–	15	17	предоставление списка уведомлений для пользователя
		–	16	–	предоставление подсказок пользователю по ходу работы в системе
ROA	INA	SA	17	23,24,25,26,27,31,33	поиск запроса сделанного агентом и передача ему результатов
	AWM	SA	18	22	передача данных по загруженности агентов
	MAPT	QIP	19	4,5,6,7,8,11	перенаправляет запрос с данными по управлению проектными задачами через ПССЗ
	ADPS	RIP	20	2,3,9,10,12	перенаправляет запрос с данными по управлению проектными решениями на разных стадиях разработки
	SEA	QIP	21	1	перенаправляет запрос с данными для поиска проектного решения

AWM	ROA	QIP	22	–	посылает запрос на формирование загруженности агентов
MAPT	ROA	SA	23	19	передача результатов по управлению проектными задачами через ПССЗ
ADPS	ROA	SA	24	20	передача результатов анализа VHDL кода
		SA	25	20	передача результатов формирования СФЛМ
		SA	26	20	передача результатов формирования шаблона СФЛМ
		SA	27	36	передача результатов сохранения/изменения проектного решения
	ASPS	QIP	28	20	передача данных о проектных решениях, которые необходимо объединить
	AKB	RIP	29	20	передача запроса поиска проектных решений на определенной стадии разработки
		RIP	30	20	передача запроса с данными для добавления, редактирования и удаления проектных решений на определенной стадии разработки
ASPS	ROA	SA	31	38	передача результата синтеза проектных решений
	AKB	RIP	32	28	передача данных для сохранения синтезируемого проектного решения в базе данных
SEA	ROA	SA	33	39	передача результатов выполнения запроса

	AKB	QIP	34	21	передача сформированного поискового запроса
		QIP	35	–	передача данных для кластеризации проектных решений
AKB	ADPS	SA	36	30	передача результатов выполнения добавления, изменения, удаления проектных решений
		SA	37	29	передача результатов выполнения поискового запроса к базе данных
	ASPS	SA	38	33	передача результатов сохранения синтезируемого проектного решения
	SEA	SA	39	34	передача результатов выполнения поискового запроса к базе данных
		SA	40	35	передача результатов кластеризации проектных решений

Описанные сценарии позволяют представить процесс взаимодействия агентов при совместном решении задач. Они позволяют сформировать набор входных и выходных данных, которые необходимы агентам, и на их основе создать интерфейсную часть многоагентной системы.

2.4. Управление процессом распределенного проектирования

В СРП возникает задача управления потоками проектных работ. Ниже предлагается ассоциативно-ориентированная модель управления задачами (работами), позволяющая эффективно организовать коллективное проектирование VHDL-объектов. В качестве основы использована модель параллельного управления работами [59,64].

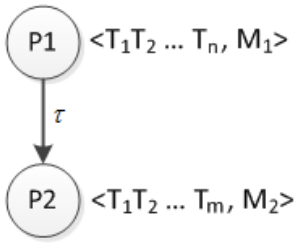
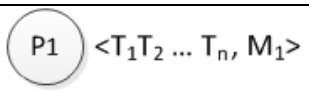
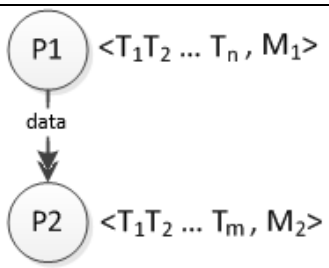
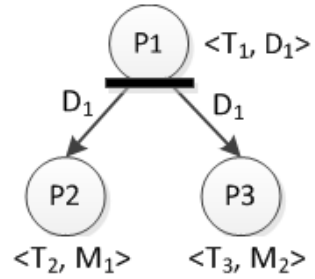
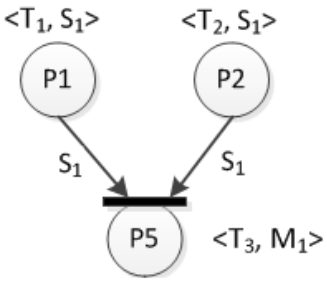
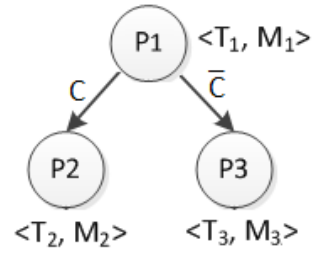
Параллельной сетевой схемой задач (ПССЗ) будем называть кортеж $PNST = \left(G(P,A), COM, CON, F, N\right)$, где $G(P,A)$ – граф ПССЗ, в котором P –

множество вершин-операторов, A – множество дуг, COM - множество составных операторов и CON – множество условных операторов. В рассматриваемом графе выделяются вершины выполнения P^E, декомпозиции P^D и синтеза P^S. В множестве составных операторов выделяются операторы проектных задач COM^T, декомпозиции COM^D и синтеза COM^S. В рассматриваемом графе выделяются дуги разветвления A^C, декомпозиции A^D и синтеза A^S. Отображение вершин графа на множество составных операторов имеет вид $F = N \to COM$. В множестве условных операторов выделяются условные операторы проектных задач CON^T, декомпозиции CON^D и синтеза CON^S. Отображение дуг графа на множество условных операторов имеет вид $N = A \to CON$. В составных и условных операторах используются элементы множеств проектных задач (T), условий переходов к следующим операторам (C), а также безусловный переход (τ) и маска (M) [71]. Составной оператор проектных задач представляет собой совокупность задач, выполнение которых происходит последовательно. Выполнение составного оператора синтеза приводит к формированию нового проектного решения, путем объединения решений, полученных на предыдущем шаге разработки, и разрешает переход к выполнению последующих задач только тогда, когда все предыдущие будут выполнены. Выполнение составного оператора декомпозиции приводит к полуавтоматическому разбиению задачи на подзадачи. Такие задачи могут выполняться независимо друг от друга с произвольным сдвигом во времени. Обобщенный условный оператор выполнения проектных задач представляет собой совокупность условий и маски.

Использование операции маскирования операторов позволяет организовать поиск по необходимым параметрам и контексту как проектных задач, так и проектных решений.

Установка связей между проектными задачами и операторами в ПССЗ происходит согласно стандарту IDEF3 [103]. Выделяются шесть типов связей, обозначения, наименование и описание которых, приведены в табл. 2.2.

Таблица 2.2. Типы связей между задачами и операторами ПССЗ

Наименование связи	Вид связи	Описание связи
Предшествование		Выполнение задач второго оператора начинается после завершения всех задач первого оператора
Отношение		Задачи выполняются параллельно
Потоков объектов		Выполнение задач второго оператора начинается после завершения всех задач первого оператора. При этом результатом работы первого оператора является объект. Эта связь также означает, что объект, порождаемый первым оператором, используется в последующих работах
Декомпозиция		Устанавливается между оператором декомпозиции и задачами, полученными в результате выполнения этого оператора.
Синтез		Устанавливается между оператором синтеза и задачами, для которых необходимо провести процедуру синтеза с целью получения проектного решения.
Условное предшествование		Выполнение задач второго оператора начинается не только после завершения всех задач первого оператора, но и выполнения определенного условия

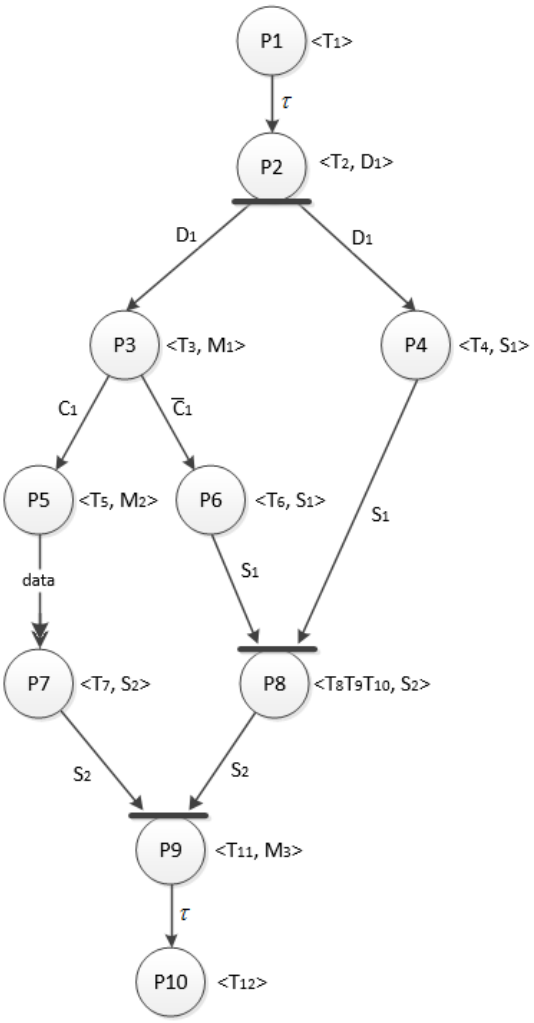


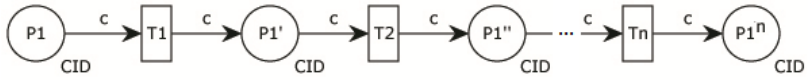
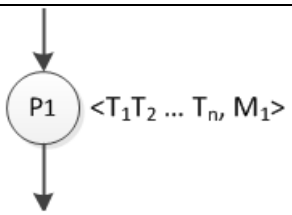
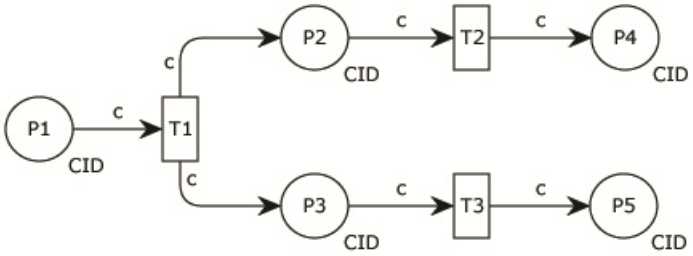
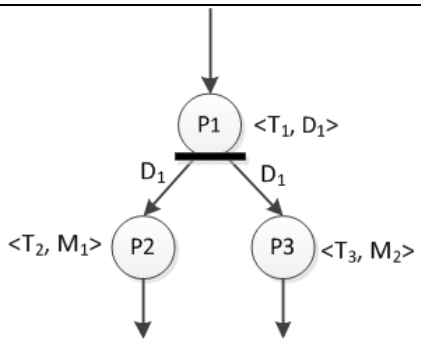
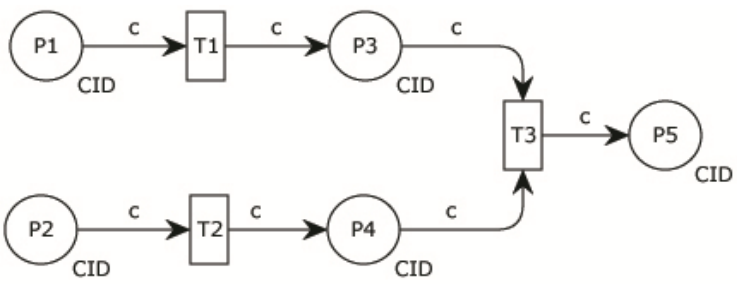
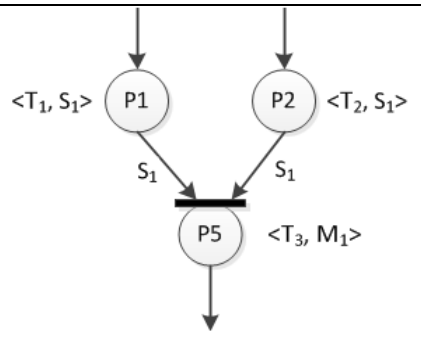
Рис. 2.9. Пример графа ПССЗ

Таким образом, передача управления в ПССЗ может быть организована одновременно по многим выходным дугам операторной вершины декомпозиции с последующим появлением условий инициирования процесса выполнения составного оператора задач. При наличии возможности передачи управления одновременно по нескольким входным дугам операторной вершины синтеза также запускается процесс выполнения составного оператора задач. На рис. 2.9. приведен пример ПССЗ содержащей все шесть типов связей, которые были описаны в табл. 2.2.

Признаком ассоциации для ПССЗ является, например, пара (P_2, D_1), а значением ассоциации - пара (P_4, $<T_4S_1>$). Ассоциативность позволяет по набору признаков отобрать проектные задачи и их решения, в описании которых присутствуют эти признаки. Способ хранения информации в ассоциативной сети таков, что наиболее часто используемые данные (проектные задачи и решения) оказываются и наиболее доступными. По типу образования в ПССЗ преобладают причинно-следственные ассоциации [64].

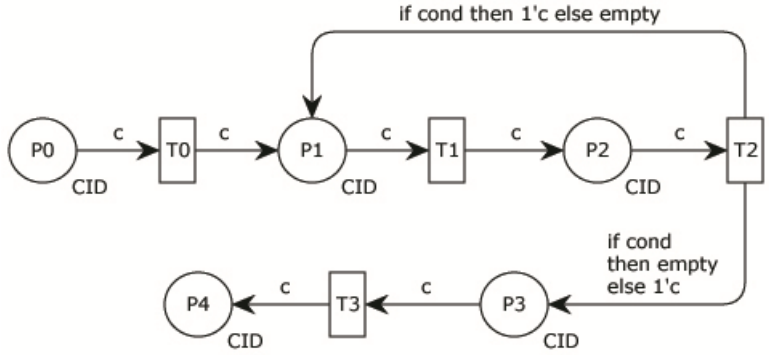
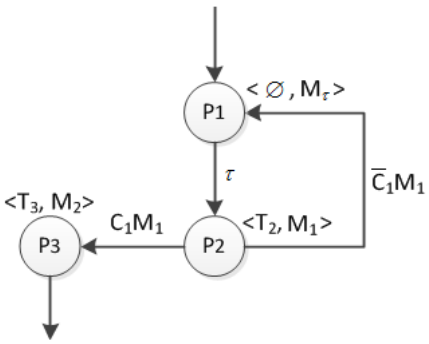
Формирование операторов ПССЗ производится на основе текста технического задания, содержащего описания параметров проектных задач или спецификаций проектируемого устройства. При этом используются шаблоны управления, показанные в табл. 2.3. Представление шаблонов в виде сетей Петри описаны в [43].

Таблица 2.3. Шаблоны ПССЗ

1. Последовательность	
Организует процесс проектирования, в ходе которого задача автоматически переходит в статус исполняемой после завершения предыдущей задачи.	
Сеть Петри	ПССЗ
	
2. Декомпозиция	
Организует процесс проектирования, в ходе которого проектная задача может быть разбита на 2 и более подзадач, выполнение которых будет происходить параллельно.	
Сеть Петри	ПССЗ
	
3. Синтез	
Организует процесс проектирования, в ходе которого возникает необходимость объединить проектные решения, полученные на предыдущем шаге проектирования, перед тем как приступить к выполнению следующей задачи.	
Сеть Петри	ПССЗ
	

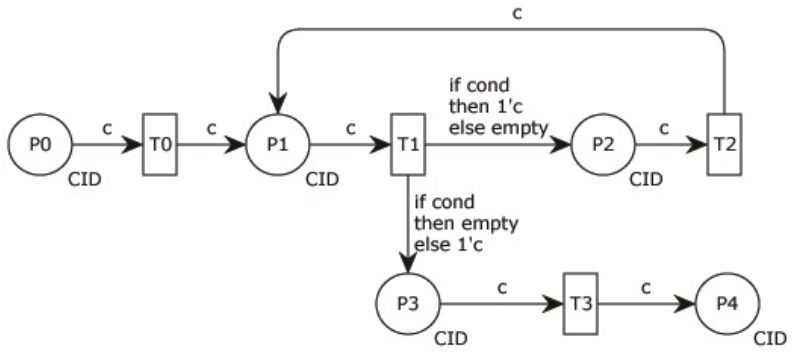
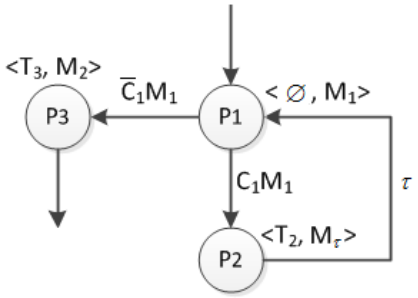
4. Цикл с постусловием

Организует процесс проектирования, в ходе которого возникает необходимость вести итерационный процесс разработки с последующей проверкой на корректность работы по определенным условиям.

Сеть Петри	ПССЗ
	

5. Цикл с предусловием

Организует процесс проектирования, в ходе которого возникает необходимость задавать произвольный интервал повторения типовых задач.

Сеть Петри	ПССЗ
	

Модель управления в виде ПССЗ классифицирует множество проектных задач описывающих проектные ситуации по контексту и признакам. Улучшение учета критериев, параметров и специфики процесса проектирования, сложных VHDL-объектов осуществляется за счет применения параллельно-последовательного управления проектными задачами.

2.5. Формальное определение многоагентной системы распределенного проектирования

Формальное определение многоагентной СРП имеет следующий вид:

$MAS = (Ag, R, En)$, где множество агентов Ag состоит из подмножеств агентов, относящихся к разным типам.

$Ag = Ag_{INA} \cup Ag_{MAPT} \cup Ag_{ADPS} \cup Ag_{ASPS} \cup Ag_{ROA} \cup Ag_{SEA} \cup Ag_{AKB} \cup Ag_{AWM}$,

$R = \{R_1, R_2, ..., R_M\}$ – множество взаимодействий между агентами,

$En = \{Dev, T, SFLM, PNST\}$ – внешняя среда,

$Dev = \{Dev_1, Dev_2, ..., Dev_F\}$ – множество проектировщиков,

Работая в системе СРП, проектировщик совершает определенные действия. $ActDev = \{ActDev_1, ActDev_2, ..., ActDev_L\}$ – множество действий проектировщика,

$T = \{T_1, T_2, ..., T_L\}$ – множество проектных задач в СРП,

$T_j = \{Dev, Name, Descr, Time, Prj, GitBranch\}$ – проектная задача,

$Name$ – название проектной задачи,

$Descr$ – описание проектной задачи,

$Time$ – время реализации проектной задачи,

Prj – привязка к проекту,

$GitBranch$ – ветка разработки проектной задачи в системе контроля версий,

$SFLM = \{SFLM_1, SFLM_2, ..., SFLM_G\}$ – множество структурно-функциональных лингвистических моделей,

$SFLM_i = \{Obj, Map\}$ – структурно-функциональная лингвистическая модель,

$Obj = \{Obj_1, Obj_2, ..., Obj_V\}$ – множество объектов проектируемого устройства.

$Obj_j = \{Dec, Proc\}$ – объект структурно-функциональной лингвистической модели, где Dec – декларативная часть, $Proc$ – процедурная часть.

Map – карта назначения входных и выходных сигналов проектируемых устройств.

$PNST = (G(P, A), COM, CON, F, N)$ – параллельно-сетевая схема задач, которая была описана в параграфе 2.4.

$COM = \{COM_1, COM_2, ..., COM_X\}$ – множество составных операторов.

$COM_j = \{T, OpList_{prev}, OpList_{next}, Type, Mask\}$ – составной оператор ПССЗ.

$CON = \{CON_1, CON_2, ..., CON_F\}$ – множество условных операторов.

$CON_j = \{T, OpList_{prev}, OpList_{next}, Type, Cond, Mask\}$ – условный оператор ПССЗ,

$OpList_{prev} = \{Op_1, Op_2, ..., Op_Z\}$ – список предыдущих операторов,

$OpList_{next} = \{Op_1, Op_2, ..., Op_Z\}$ – список следующих операторов,

$Op_j \in (COM \mid CON)$.апра

$Type = \{task, decomposition, synthesis\}$, где *task* – проектные задачи, *decomposition* – декомпозиция, *synthesis* – синтез.

Cond – условие при котором будет осуществлен переход к следующему оператору. В случае безусловного перехода $Cond = \tau$.

$Mask = \{CodeM_1, CodeM_2, ..., CodeM_T\}$ – маска оператора, описывает атрибуты проектных задач, которые участвуют/не участвуют в поиске.

$$CodeM_i = \begin{cases} 0, \text{ не участвует в поиске} \\ 1, \text{ участвует в поиске} \end{cases}.$$

В виду того, что в СРП выделяются ряд модулей, не являющиеся агентами, но которые предоставляют возможность взаимодействовать проектировщику с самой системой, введем в рассмотрение формальное определение СРП: $SDD = \{SysSt, In, Out, Act, DB\}$.

$SysSt = \{SysSt_1, SysSt_2, ..., SysSt_O\}$ – множество состояний, которые характеризуют ключевые этапы получения проектных решений через модули СРП.

In – запрос к СРП,

Out – сформированный ответ на запрос,

DB – база данных СРП,

$Act = \{Act_1, Act_2, ..., Act_P\}$ – множество внутренних действий СРП,

Математическая модель агента многоагентной системы имеет следующий вид: $Ag = \{In_{Ag}, Out_{Ag}, AgSt, ActAg\}$.

$AgSt = \{AgSt_1, AgSt_2, ..., AgSt_K\}$ – множество возможных состояний агента,

$ActAg = \{ActAg_1, ActAg_2, ..., ActAg_R\}$ – множество действий агента,

$ActAg_i = \{name, reciever, type\_mess\}$, где *name* – наименование действия, *receiver* – получатель, *type_mess* – тип отправляемого сообщения.

$In_{Ag} = \{Mes_{acl}, Query\}$ – рецепторный вход агента, который может быть в виде запроса ($Query$) или сообщения ACL (Mes_{acl}).

$Out_{Ag} = \{Mes_{acl}, Request\}$ – эффекторный выход агента, который может быть в виде ответа ($Request$) или сообщения ACL (Mes_{acl}).

Рассматривая основные сущности СРП, такие как проектировщики, компоненты СРП, агенты, база проектных решений СФЛМ и ПССЗ, выделим следующие взаимодействия между ними.

1. Между проектировщиком и компонентами СРП.

$Dev \times T \times SysSt \rightarrow ActDev \times SysSt \times [Init(Ag) = In_{Ag}]$ – обращение проектировщика к СРП, где $Init(Ag)$ – функция инициализации агента путем формирования и передачи данных на его вход In_{Ag}.

$InData = (Find(Dev, DB) \mid Pars(In))$, где функция $Find(Dev, DB)$ ищет сообщения для проектировщика в базе данных СРП, а функция $Pars(In)$ обрабатывает входные данные.

$SysSt \times InData \rightarrow Act \times Out \times SysSt$ – внутрисистемное функционирование.

$SysSt \times InData \times Out \rightarrow (Provide(Out, Dev) = \{view, false\}) \times SysSt$, где функция $Provide(Out, Dev)$ предоставляет данные проектировщику.

Рассматривая более детально процесс внутрисистемного функционирования по концепции MVC, предыдущая формула преобразуется следующим образом:

$SysSt \times (InData_{controller} \times InData_{method} \times InData_{data}) \rightarrow Act_{model} \times (Out_{controller} \times$
$\times Out_{method} \times Out_{data}) \times SysSt,$

$SysSt \times (InData_{controller} \times InData_{method} \times InData_{data}) \times (Out_{controller} \times Out_{method} \times$
$\times Out_{data}) \rightarrow (Provide(Out_{data}, Dev) = Out_{view}) \times SysSt.$

2. Между агентами, базой проектных решений и ПССЗ.

Внешняя среда агента представлена окружающими его объектами, в том числе и другими агентами. На рецепторный вход агента поступают сообщения, которые необходимо разобрать с помощью функции $Pars(In_{Ag})$ и получить необходимые данные, или агент может сформировать входные данные сам на основе текущего состояния через функцию $Create(AgSt)$.

$In_{Ag\_data} = (Create(AgSt) \mid Pars(In_{Ag}))$ – формирование входных данных.

Функционирование агента описывается следующим образом:

$AgSt \times In_{Ag\_data} \times \lfloor FuncDB\_SFLM(In_{data}, DB_{SFLM}) = DB_{SFLM\_RES} \rfloor \times$
$\times \left[FuncPNST(In_{Ag\_data}, PNST) = PNST_{RES} \right] \times \left[Change(In_{Ag\_data}) = Data \right] \rightarrow$
$\rightarrow ActAg \times Out_{Ag} \times AgSt,$

где DB_{SFLM} – база проектных решений СФЛМ, функция $Change(In_{Ag\_data})$ преобразует входные данные в соответствии с внутренним состоянием агента, $FuncDB\_SFLM(In_{data}, DB_{SFLM})$ представляет собой совокупность функций, название и результат работы (DB_{SFLM_RES}) которых представлены в табл. 2.4, $FuncPNST(In_{Ag\_data}, PNST)$ представляет собой совокупность функций, название и результат работы ($PNST_{RES}$) которых представлены в табл. 2.5.

Таблица 2.4. Описание функций по работе с СФЛМ и их результатов

$FuncDB\_SFLM$	DB_{SFLM_RES}
Create	{true, false}
Find	{SFLM, false}
Update	{true, false}
Delete	{true, false}

Таблица 2.5. Описание функций по работе с ПССЗ и их результатов

FuncPNS	$PNST_{RES}$
CreateOperator	{true, false}
FindTask	{Op, false}
FindPNST	{PNST, false}
FindDev	{Dev, false}
UpdateOperator	{true, false}
DeleteOperator	{true, false}

Опишем возможные данные на рецепторных входах агентов.

1. В случае когда на вход агента поступает запрос:

$In_{Ag\_data}: \{uid, action, controller, content\}$, где *uid* – id пользователя, *action* – действие, *controller* – контроллер, *content* – содержимое.

2. В случае когда на вход агента поступает acl сообщение:

$In_{Ag\_data}: \{uid, sender, reciever, content, reply\_with, reply\_to, envelope,$
$language, ontology, reply\_by, protocol, conversation\_id\}$,

где *uid* – идентификатор пользователя в системе; *sender* – идентификатор агента-отправителя сообщения; *receiver* – кортеж идентификаторов агентов-получателей; *content* – содержимое сообщения или объект действия; *reply_with* – выражение, которое будет использоваться агентом, чтобы определить исходное сообщение при ответе на полученное сообщение; *reply_to* – выражение, ссылающееся

на раннее действие, к которому текущее сообщение является ответом; *envelope* – конверт, содержащий полезную информацию о сообщении, представляет собой список пар значений ключевых слов; *language* – обозначает кодировку содержимого; *ontology* – онтология, которая используется, чтобы дать смысл содержимому; *reply_by* – крайний срок отправки ответа на сообщение в виде времени и/или даты; *protocol* – идентификатор протокола, который использует агент-отправитель (протокол предназначен для формирования дополнительного контекста, служащего для интерпретации сообщения); *conversation_id* – выражение для идентификации последовательности коммуникативных актов, которые вместе образуют «разговор» [12].

3. Между агентом и СРП:

$AgSt \times In_{Ag\_data} \times Out_{Ag} \rightarrow (SaveDB(Out_{Ag}, DB) = \{true, false\}) \times AgSt$,где

$SaveDB(Out_{Ag}, DB)$ – функция сохранения в базе СРП данных, полученных агентом, которые необходимо предоставить проектировщику.

На основе формульных зависимостей была разработана структура моделей в виде цветной сети Петри и назначения ее позиций и переходов. Процесс получения проектного решения начиная с формирования проектной задачи описывается совокупностью сценариев: вход в СРП, поиск проектных задач, поиск проектировщика, поиск проектного решения, создание и верификация проектного решения, формирование модели СФЛМ, интеграция проектных решений от проектировщиков, информирование проектировщиков. Данные сценарии представлены в виде моделей цветной сети Петри.

2.6. Моделирование работы системы

Описание и анализ процессов, протекающих в системе, осуществляется средствами математического аппарата цветных сетей Петри второго рода, представленных в виде иерархической композиции объектов [68].

Сети Петри за счет свойства параллелизма или одновременности представляются наиболее подходящими для моделирования СРП, в которой осуществляется распределенное управление проектными задачами и создание проектных решений.

Анализ сетевой модели до разработки программного обеспечения системы проектирования обеспечивает проверку работоспособности, выявления ошибок и узких мест в системе, из-за которых производительность или пропускная способность системы ограничена одним или несколькими компонентами или ресурсами.

Модель СРП реализована в среде CPN Tools [78,125] и показана на рис. 2.10, в табл. 2.6 приведено описание позиций и переходов.

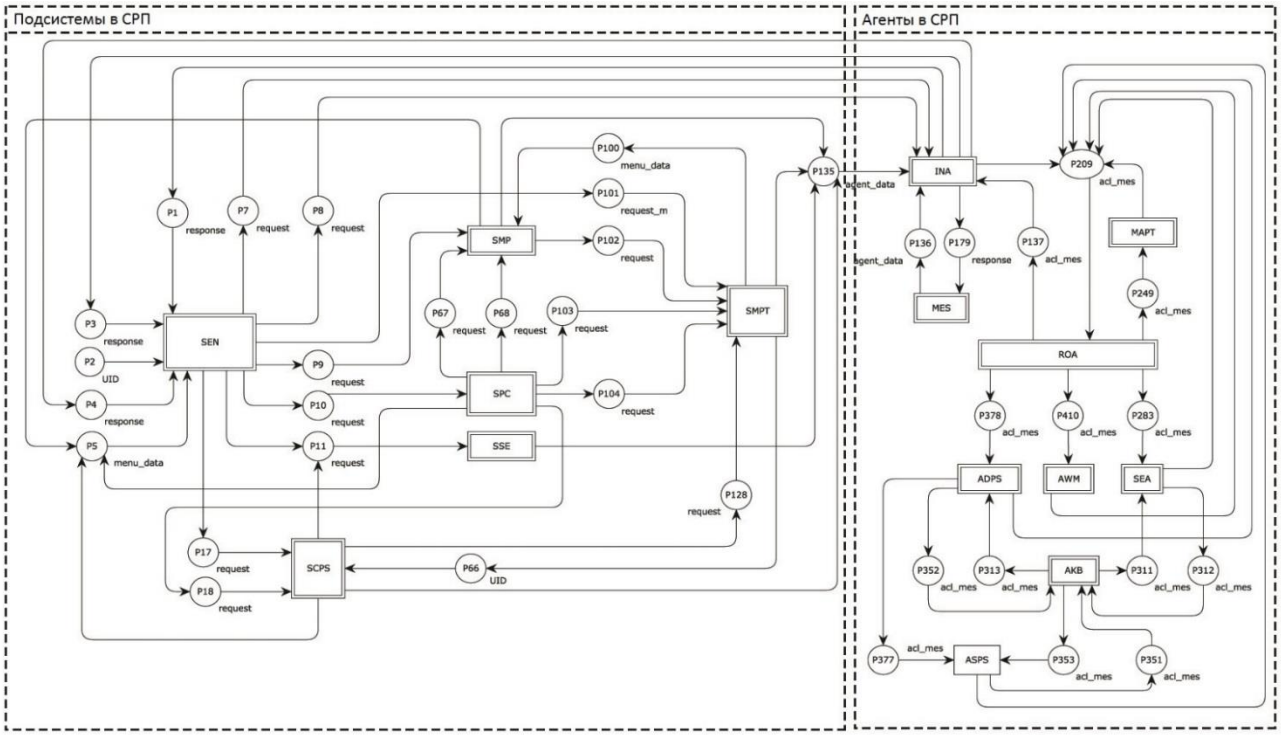


Рис. 2.10. Сеть Петри системы распределенного проектирования

Выделены следующие блоки, моделирующие процесс работы проектировщиков в системе: system entry (SEN) – система входа; system search (SSE) – система поиска; system constructor project solution (SCPS) – система конструктора проектного решения; system personal cabinet (SPC) – система личного кабинета; system managing projects (SMP) – система управления проектами; system managing project tasks (SMPT) – система управления проектными задачами; messenger (MES) – система уведомлений.

Таблица 2.6. Описание позиций и переходов сети Петри СРП

Обозначение элемента	Описание
P1	Были выявлены ошибки при авторизации пользователя в системе
P2	Были выявлены ошибки при регистрации пользователя в системе
P3	Осуществлен переход на стартовую страницу системы
P4	Авторизация в системе пройдена. Осуществлен вход в систему
P5	Выбран элемент основного меню
P7	Передан запрос на регистрацию пользователя в системе
P8	Передан запрос на авторизацию пользователя в системе
P9	Осуществлен переход на страницу управления проектами
P10	Осуществлен переход в личный кабинет
P11	Осуществлен переход на страницу поиска
P17	Осуществлен переход к конструктору проектного решения

P101	Осуществлен переход к управлению проектными задачами
P135	Передан запрос интерфейсному агенту
P67	Осуществлен переход на страницу создания проекта
P68	Выбран проект. Осуществлен переход к странице проекта
P104	Осуществлен переход к проектной задаче
P103	Осуществлен переход к странице управления результатами поиска ПССЗ
P18	Осуществлен переход к странице управления результатами поиска СФМ
P100	Осуществлен переход к меню. Выбран элемент меню
P102	Осуществлен переход к просмотру списка проектных задач с фильтрацией по проекту и/или проектировщику
P66	Осуществлен переход к странице создания/редактирования проектного решения
P128	Осуществлен переход на страницу создания проектной задачи
P179	Переданы данные по уведомлениям для пользователя
P136	Передан запрос на получение уведомлений для пользователя
P137	Передано сообщение от агента ROA. Сообщение получено
P209	Сформировано и передано сообщение агенту ROA
P249	Передано сообщение от агента ROA. Сообщение получено
P283	Передано сообщение от агента ROA. Сообщение получено
P311	Передано сообщение от агента AKB. Сообщение получено
P312	Сформировано и передано сообщение агенту AKB
P351	Передано сообщение от агента ASPS. Сообщение получено
P352	Передано сообщение от агента ADPS. Сообщение получено
P378	Сформировано и передано сообщение агенту ADPS
P353	Сформировано и передано сообщение агенту ASPS
P377	Передано сообщение от агента ADPS. Сообщение получено
P378	Передано сообщение от агента ROA. Сообщение получено
P410	Передано сообщение от агента ROA. Сообщение получено

В качестве примера на рис. 2.11 и в табл. 2.7 представлена подсистема SEN.

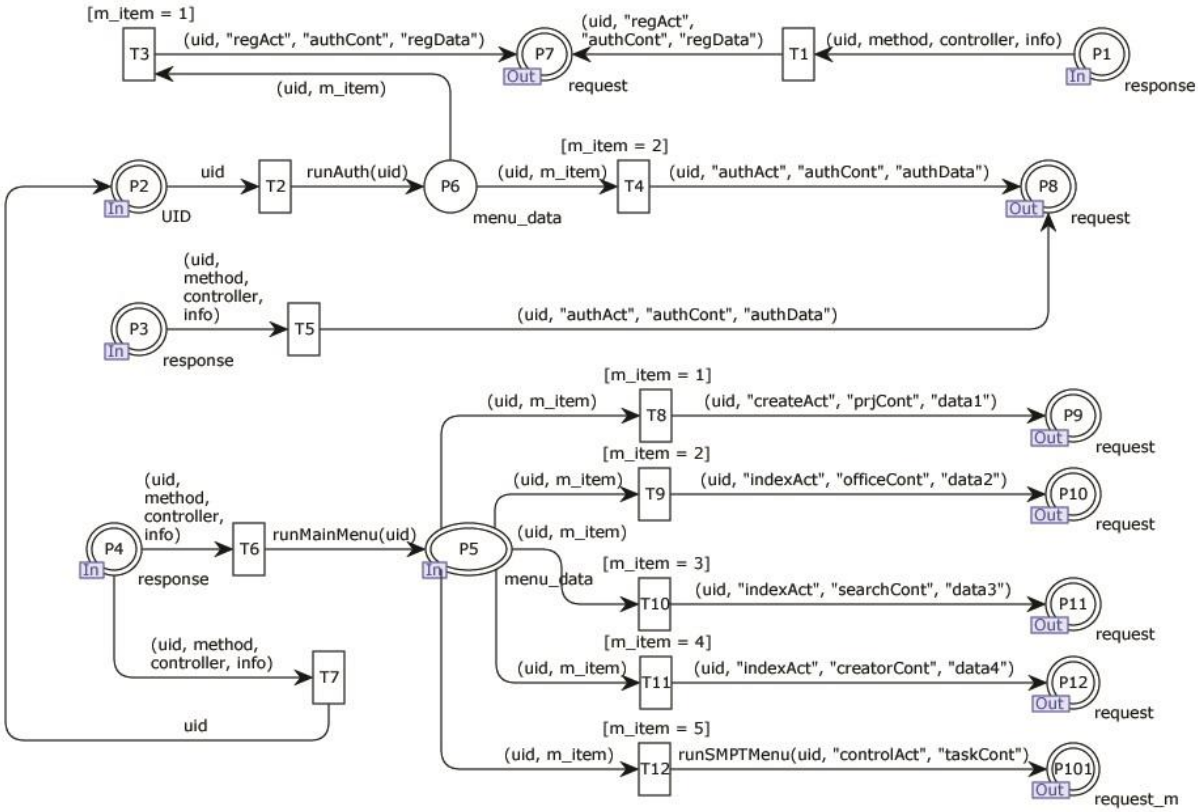


Рис. 2.11. Сеть Петри подсистемы SEN

Таблица 2.7. Описание сети Петри подсистемы SEN

Обозначение элемента	Описание
P1	Ошибки в регистрационных данных
P2	Осуществлен вход на стартовую страницу системы
P4	Авторизация успешна. Осуществлен вход в систему
P5	Выбран элемент основного меню
P3	Ошибки в данных авторизации
P7	Отправлен запрос на регистрацию
P8	Отправлен запрос на авторизацию
P9	Выполнен переход на страницу создания проекта
P10	Выполнен переход в личный кабинет
P11	Выполнен переход на страницу поиска
P12	Выполнен переход в конструктор проектного решения
P6	Ожидание авторизации / регистрации
P101	Выполнен переход на страницу управления проектными задачами
T1	Исправление ошибок в данных регистрации
T2	Вход в систему
T3	Регистрация и вход в систему
T4	Авторизация и вход в систему
T5	Исправление ошибок в данных авторизации
T6	Выбор элемента основного меню

T7	Выход из системы
T8	Переход на страницу создания проекта
T9	Переход в личный кабинет
T10	Переход на страницу поиска
T11	Переход в конструктор проектного решения
T12	Переход на страницу управления проектными задачами

Сетевые модели остальных подсистем, а также агентов, представлены в приложениях 1-14.

Моделирование работы проектировщиков в системе выполняется путем создания токенов в стартовой позиции сети Петри через случайные интервалы времени. Действия проектировщиков моделируется путем запуска переходов в сети, описывающей подсистемы СРП. Остальные события в сети представляют собой реакцию интерфейсных агентов на запрос проектировщиков и взаимодействие агентов между собой. Основными типами токенов являются сообщения взаимодействия агентов на языке Agent Communication Language (ACL) а также запросы проектировщиков к СРП. Данные типы были описаны в параграфе 2.5.

К позициям сети Петри добавляется информация о типах токенов, которые могут находится в данном месте. В моделях преобладающими типами токенов являются acl_mes и request.

К дугам, исходящим из позиций сети, добавляется информация, которая может быть представлена в виде типа токена или совокупности атрибутов, этого типа токена. Таким образом, переданные данные могут участвовать в возбуждении переходов, инцидентных этим дугам.

За счет добавления такой информации в качестве предиката возбуждения перехода можно осуществить проверку на наличие внутреннего правила работы агента или системы и выполнение в соответствии с результатом проверки действия. Внутренние правила работы агента основаны на обработке ACL от других агентов. Правила работы системы, построенной по технологии MVC (Model View Controller), основаны на проверке наличия в системе вызываемого контроллера и метода.

Формирование токена для выходной позиции происходит с учетом выражения на дугах. Таким образом, переход может порождать токены нового типа путем объединения данных [102]. В описываемой системе такое преобразование происходит в том случае, когда данные типа токена request участвуют в формировании acl_mes с сохранением определенных данных, таких как идентификатор пользователя сделавшего запрос.

К начальной маркировке сети добавляется информация о значениях переменных, содержащихся в токенах [67].

Цветная сеть Петри представлена четырьмя элементами: $CPN = (G, C, F, M_0)$, где $G = (P, T, I, O)$ – биграф, в котором

$P = \{p_i\}$, где $i = 1,2,...,N$ – конечное множество позиций,

$T = \{t_j\}$, где $j = 1,2,...,M$ – конечное множество переходов,

$I : T \to P$ – входная функция-отображение множества переходов во множество позиций. Результат функции – дуга *A* выходящая из перехода *T* в позицию *P*.

$O : P \to T$ – входная функция-отображение множества позиций во множество переходов. Результат функции – дуга *A* выходящая из позиции *P* в переход *T*.

Функции I и O задают множества дуг $\{t_j, p_i\}$ и $\{p_i, t_j\}$.

$C = \{X, R\}$ – пара, описывающая множество цветов и их отношения, где $X = \{\lambda, c_1, c_2, ..., c_L\} = \{c_r\}$ – множество раскрашивающих цветов или других признаков, включающее элемент λ, обозначающий «отсутствие цвета» *R* - бинарное отношение на множестве цветов *X* (чаще всего это отношение эквивалентности или равенства цветов).

M_0 начальное маркирование (состояние) сети;

Позиции и переходы связываются с помощью дуг *A*. $F : A \to X$ отображение множества дуг биграфа *G* на множество цветов, дающее раскраску дуг: c_{ij}^S - цвет s-ой дуги a_{ij}^S биграфа, направленной из *i* позиции в *j* переход.

Цветное маркирование будем представлять в следующем виде: $M = \{m_i\}$ – конечное множество маркировок в позициях сети, где $m_i = \{m_i^1, m_i^2, ..., m_i^r, ..., m_i^L\}$, m_i^r – число меток r-го цвета в позиции p_i.

Охарактеризовать цветное маркирование можно двумя функциями:

$V : P \to \mu$ – функция размещения меток по позициям, $Z : \mu \to X$ – функция размещения цветов по меткам, где μ – множество цветных меток.

Факт размещения метки m в позиции p_i обозначим $m \in p_i$

Факт наличия цвета у метки будем представлять в следующем виду: $c(m)$

Входные позиции перехода t_j объединяются в множества его предшественников $\Pr(t_j) = \{p_i \in P \mid O(p_i, t_j) = 1\}$, а выходные позиции – в множества позиций–последователей $Post(t_j) = \{p_i \in P \mid I(t_j, p_i) = 1\}$. Запись $I(t_j, p_i) = 1$ означает наличие дуги (t_j, p_i), а запись $O(t_j, p_i) = 1$ – дуги (p_i, t_j).

Логическое уравнение, описывающее условие возбуждения перехода t_j, имеет следующий вид: $\forall p_i \in \Pr(t_j)\ \exists\ m \in p_i [R(c(m), c_{ij}^S) = 1] \to u(t_j) = 1$, иначе $u(t_j) = 0$.

В цветных сетях Петри у перехода может быть задано условие срабатывания, которое зависит от цветов дуг входящих в этот переход. Таким образом, логическое уравнение, описывающее условие возбуждения перехода t_j, примет следующий вид: $[\forall p_i \in \Pr(t_j)\ \exists\ m \in p_i [R(c(m), c_{ij}^S) = 1]] \& [Z(t_j) = 1] \to u(t_j) = 1$, иначе $u(t_j) = 0$, где $Z(t_j)$, функция описывающая условие срабатывания перехода.

Если переход t_j возбужден ($u(t_j) = 1$), то происходит его срабатывание, и возбуждавшие метки из позиций $Pr(t_j)$ при этом изымаются, а позиции $Post(t_j)$ пополняются метками, цвет которых принимается равным цвету дуг, проходящих из t_j в позиции $Post(t_j)$.

Исследование сети Петри производится по двум направлениям:

1) Направление исследования функционирования всей цветной сети Петри;

2) Направление исследования функционирования цветной сети Петри, связанное с изменением количества фишек в конкретной позиции в процессе функционирования сети.

Анализ заключается в изучении основных свойств сетей Петри и описание возможного их применения как для проверки работоспособности системы до ее создания, так и в процессе ее функционирования [95]. В цветных сетях Петри анализ проводится с учетом цвета токена, что позволяет моделировать коллективную работу проектировщиков, имеющих различные токены-идентификаторы.

К первому направлению относится анализ сети на свойство живости. Результат анализа показывает степень работоспособности, помогает выявлять невозможные состояния в моделируемой системе (например, неисполняемые ветви функционирования системы), места, которые блокируют пользователя на одной странице в web-системе или работа в которых зацикливается на каком-то действии. В цветной сети Петри анализ этого свойства производится относительно каждого цвета (т.е. относительно каждого проектировщика, находящегося в системе). Таким образом, осуществляется проверка работы в системе нескольких проектировщиков, выполняющих различные действия. Подобное моделирование очень приближено к реальному использованию системы.

Цветная сеть Петри считается живой, если выполняется следующие условия:

$$\exists M' \in H(CPN, M_0) \exists t_j \forall p_i \in \Pr(t_j) \ \exists \ m \in M' \& \ m \in p_i [R(c(m), c_{ij}^S) = 1]] \& \\ \& [Z(t_j) = 1]$$

$\exists M'' \neq M'$.

Также к первому направлению относится анализ сети на достижимость.

При моделировании работы системы задача достижимости интерпретируется как возможность перехода к некоторой типовой ситуации, под которой понимается совокупность последовательных маркировок, необходимых для перехода в интересующее состояние системы. Для этого строится дерево типовых ситуаций, переходы по которому переводят систему в выбранные состояния, минуя определенное количество промежуточных этапов. Используя свойство достижимости сети Петри, строится карта внутрисистемных переходов. На ней отображаются все переходы, но активны будут только те, в которые можно перейти из текущего состояния.

Задача достижимости заключается в определении для маркировки M_0 маркировки $M \in H(CPN, M_0)$, где H – функция, возвращающая множество доступных маркировок в сети CPN из M_0.

Задача покрываемости. Для сети Петри с начальной маркировкой M_0 и маркировки M определить, существует ли такая достижимая маркировка $M' \in H(CPN, M_0)$, что $M' \geq M$.

Ко второму направлению относится анализ сети на свойство безопасности. Определение свойства безопасности позволяет анализировать процесс создания проектного решения «наблюдением» за ходом выполнения проектных задач, выполняемых проектировщиками. Так как в текущий момент у проектировщика может быть только одна активная задача, которую он делает и по которой ведется учет времени, анализ свойства безопасности позиций токенов определенного цвета позволит выявить временные нарушения решения задач.

Для цветной сети Петри CPN с начальной маркировкой M_0 позицию $p_i \in P$ является безопасной относительно цвета c_r если для любой маркировки, M' достижимой из M_0, выполняется $M'^{Cr}(p_i) = 1$:

$$\exists\ p_i \in P \forall M' \in H(CPN, M_0)[m_i^r = 1] \rightarrow p_i - безопасна.$$

Обязательное наличие позиции-очереди у агента создает необходимость проверять данную позицию на свойство ограниченности. В случае превышения предельно допустимого количества задач, установленного в системе,

запускается механизм создания копии агента и переназначения на него части задач. Освобождение ресурсов удалением копии агента из системы происходит при отсутствии задач в позиции-очереди агента.

Для цветной сети Петри CPN, позицию $p_i \in P$ назовем K-ограниченной относительно цвета c_r, если для любой маркировки M', достижимой из M_0, выполняется $M'^{Cr}(p_i) = K$.

$$\exists\ p_i \in P \forall M' \in H(CPN, M_0)[m_i^r = K] \rightarrow p_i\ K - ограничена$$

Анализ результатов может сказать о том, какие действия происходят, какие состояния им предшествовали, и какие состояния примет система после выполнения действия, в каких состояниях пребывала или не прибывала система. Таким образом, за счет маркирования, СРП снабжается историей состояний (для каждого проектировщика). Подобный функционал позволит возвращаться в предыдущие состояния для корректировки данных с последующим продолжением работы. Таким образом, формируя различные событийные линии, к которым можно будет возвращаться в процессе работы, проектировщик может сравнивать результаты, полученные в различных событийных линиях. Например, создание проектного решения путем перебора шаблонов из базы данных и изменения параметров в этих шаблонах.

Модель системы на базе аппарата сетей Петри реализована в среде CPN Tools, которая позволяет смоделировать и проанализировать работу системы с учетом ее архитектуры, выявить наиболее загруженные элементы системы [66]. Результаты моделирования в пакете CPN Tools [125] приведены в табл. 2.8.

Таблица 2.8. Результаты моделирования в пакете CPN Tools

Статистика	Комментарии
State Space Nodes: 42096 Arcs: 101652 Secs: 73 Status: Full Scc Graph Nodes: 344 Arcs: 1086	Пространство состояний модели содержит 42096 узлов и 101652 дуг

Secs: 6	
Home Markings None	Модель не имеет домашних маркировок
Dead Markings None	Модель не имеет мертвых маркировок
Dead Transition Instances None Live Transition Instances None	В модели отсутствуют мертвые переходы
Impartial Transition Instances None	Бесконечная последовательность срабатывания отсутствует

В ходе моделирования были выявлены узкие места в системе, путем оценки количества меток в определенных ключевых позициях, и были предприняты меры по реконструированию данных мест. Результатом моделирования проектируемой системы было подтверждено отсутствие мертвых маркировок и переходов, сообщая о том, что при работе проектировщика в системе не будут возникать тупиковые состояния, вынуждающие перезапускать систему, и весь функционал будет востребован в ходе распределенного проектирования.

2.7. Выводы

1. При построении современных САПР одним из основных подходов повышения эффективности их работы, в том числе и при коллективном проектировании, является повышение их интеллектуальности. Предлагаемая архитектура СРП, в основе математического обеспечения которой лежит многоагентная технология, относится к классу интеллектуальных систем и обеспечивает распределенное коллективное проектирование сложных HDL-проектов.

2. Функционирование СРП основано на сценариях взаимодействия агентов.

3. Учитывая предпосылки перехода на инновационную модель получения IT-услуг в рассматриваемой архитектуре применяется такая модель в виде частного облака, использующего бизнес-модель SaaS.

4. При необходимости организации взаимодействия в корпоративной сети особое внимание уделяется поддержке концепции распределенных объектных систем за счет применения технологии CORBA, которая используется в качестве основной при реализации СРП.

5. При распределенном проектировании важная роль отводится процессу формирования и назначения проектных задач. Для управления процессом проектирования предложена модель управления задачами в виде параллельной сетевой схемы задач.

6. Модель ПССЗ позволяет связать и организовать порядок выполнения множества проектных задач, описывающих проектные ситуации по контексту и признакам, что оптимизирует управление сложными процессами проектирования VHDL-программ, сократить затраты разработки и повысить качество проектирования.

7. Одним из путей снижения сложности разработки осуществляется за счет готовых абстракций для решения целого класса задач. Этот подход описан в виде шаблонов ПССЗ, предназначенных для упрощения процесса формирования конструкций, управляющих процессом проектирования.

8. В ходе распределенного проектирования возникают задачи, которые необходимо делегировать агентам, чтобы упростить работу проектировщиков в системе и уменьшить время, затрачиваемое на получение проектного решения. Были выделены и описаны подобные задачи, такие как, объединение проектного решения, поиск, управление процессом проектирования и др.

9. Рассматривая возможности методологии повторного использования, позволяющей решать сложные задачи, предложенная архитектура СРП содержит функциональные модули, предназначенные для коллективной разработки проекта, обеспечивая наполнение библиотек проектных решений с целью их повторного использования.

10. При разработке сложных параллельных и распределенных систем важным этапом является их моделирование, которое позволяет устранить дорогостоящие ошибки на этапе проектирования. Проведенное моделирование и

анализ работы СРП на базе цветных сетей Петри позволил получить важную информацию о структуре, поведении, узких местах как системы в целом, так и отдельных агентов.

ГЛАВА 3. РАЗРАБОТКА МЕТОДОВ СИНТЕЗА И АНАЛИЗА РАСПРЕДЕЛЕННОГО ПРОЕКТИРОВАНИЯ СФЛМ

В данной главе исследуются методы и организация работы с СФЛМ в ходе распределенного проектирования. Предлагается схема применения концепции MVC для организации СФЛМ в виде трех отдельных компонента таким образом, чтобы модификация одного из компонентов оказывала минимальное воздействие на остальные. Описывается функциональное назначение каждого компонента и приводится пример СФЛМ.

Исследуются известные элементы синтеза и анализа проектных решений, такие как описания на HDL языке, шаблоны, библиотеки элементов. Приводятся их достоинства и недостатки.

Описывается процесс совместной работы проектировщиков с применением ПССЗ и последующим объединением проектных решений СФЛМ.

В рассматриваемой главе также представляются способы распределенного проектирования СФЛМ, позволяющие организовать взаимодействие между проектировщиками, с целью получения проектных решений. А также проводится оценка эффективности применения СФЛМ.

3.1. Применение концепции MVC для СФЛМ

Применение концепции MVC (Model-View-Controller) направлено на разделение бизнес логики от данных, описывающих функционирование проектируемого устройства, и отображения проектируемого устройства от бизнес логики, чтобы проектировщики могли легко изменять отдельные функциональные части, не затрагивая другие. Таким образом, изменения, вносимые в один из компонентов, оказывают минимально возможное воздействие на другие компоненты.

Под Моделью (Model), понимается сущность, содержащая в себе функциональную логику проектируемого устройства, представляющая совокупность структуры, описаний функционирования и интерфейсов.

В обязанности Представления (View) входит отображение данных проектируемого устройства, полученных от Модели. Однако представление не может воздействовать на модель, с целью ее изменения. Основными видами представления проектного решения в СФЛМ являются: vhdl текст, xml и графический элемент.

В обязанности контроллера (Controller) входит формирование проектного решения путем выбора модели и представления на ее основе.

Таким образом, достигается результат, позволяющий менять представление данных, а именно, как они отображаются, не трогая саму Модель [151].

Процесс работы в системе с СФЛМ по концепции MVC представлен на рис. 3.1.

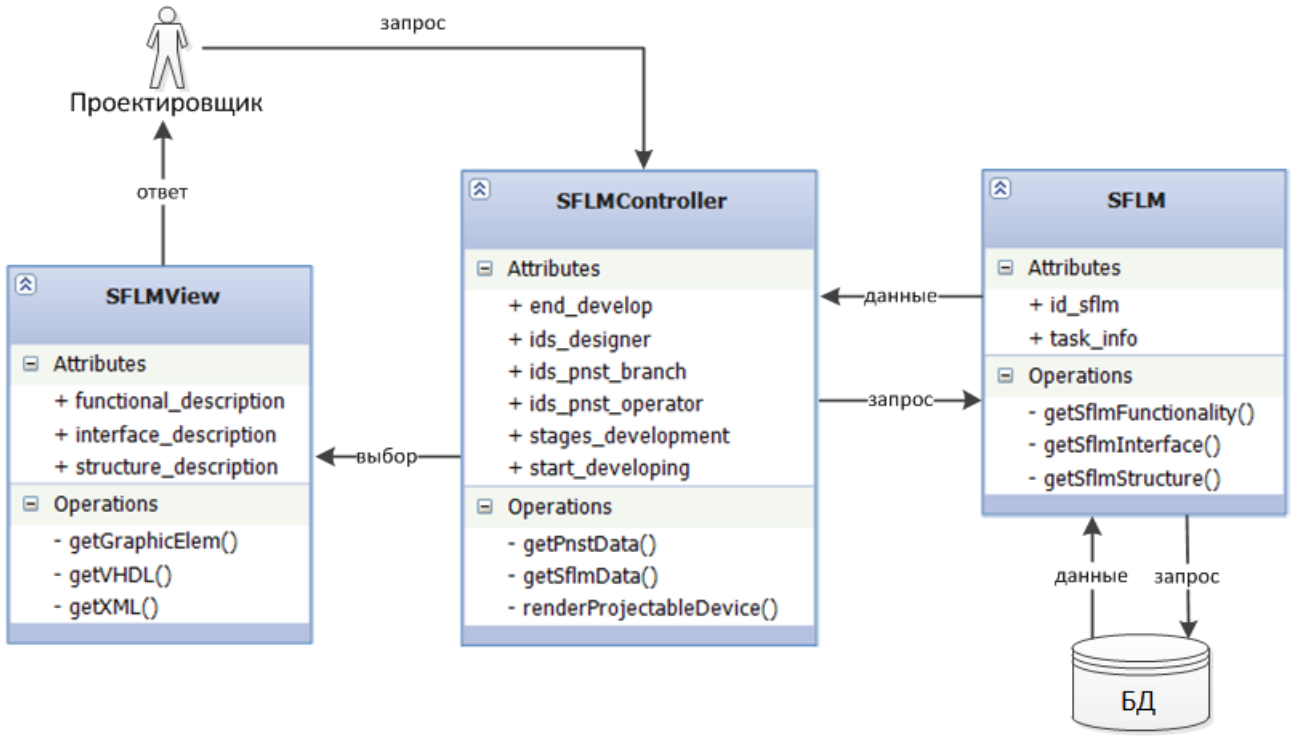


Рис. 3.1. Процесс работы с СФЛМ по концепции MVC

В контроллер (SFLMController) выносятся методы для получения данных структуры, функциональной части и интерфейса из модели (SFLM). Также контроллер отвечает за получение данных о проектных задачах из ПССЗ (Parallel network scheme tasks – PNST). В качестве поисковых параметров

рассматриваются: идентификаторы веток разработки (ids_pnst_branch); идентификаторы операторов ПССЗ (ids_pnst_operator); идентификаторы проектировщиков, которые решают проектные задачи(ids_designer); временные рамки разработки (start_developing, end_develop); стадии проектирования, такие как проектная задача, проектное решение в виде описания на языке vhdl, СФЛМ, шаблон СФЛМ, графическое представление проектного решения (stages_development).

Представление (SFLMView) отвечает за формирование вывода проектных решений в видах, один из примеров которых приведен в табл. 3.1.

Таблица 3.1. Виды представления СФЛМ для описания памяти программ (PRAM)

Представление в виде описания на языке VHDL

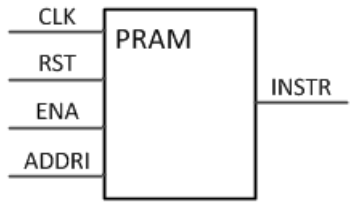
```
library IEEE;
use IEEE.std_logic_1164.all, IEEE.std_logic_UNSIGNED.all;
--pragma translate_off
entity PRAM is
        port(
                CLK : in std_logic;
                RST : in std_logic;
                ENA : in std_logic;    -- разрешение работы
                ADDRI : in WORD;     -- адрес команды
                INSTR : out WORD);     -- считанная команда
end PRAM;

architecture PRAM_BEH of PRAM is  -- поведенческая модель памяти программ
        signal address: natural;
begin
        PRAM: process
                variable PROM:PMEM_ARR;
        begin
                Load_F( PROG_FILE, PROM );  -- загрузка памяти из файла
                loop
                        if Rising_edge(CLK) and (ENA='1' or ENA='H') then
                                address<= Conv_Integer(ADDRI); -- запоминание адреса
                        end if;
                        INSTR<=PROM(address);  -- чтение команды
                        wait on CLK,RST,ADDRI,address;
                end loop;
        end process;
       end PRAM_BEH;
```

Представление в виде структуры XML

```
<vhdl_programm>
        <library>
                <name>IEEE</name>
```

```
            <package>
                <name>std_logic_1164</name>
                <element>
                    <name>all</name>
                </element>
            </package>
        </library>
        <comment>pragma translate_off</comment>
        <entity>
            <name>PRAM</name>
            <ports>
                <item>
                    <var>
                        <name>CLK</name>
                    </var>
                    <direction>in</direction>
                    <type>std_logic</type>
                </item>
                …
            </ports>
        </entity>
        <architecture>
            <name>PRAM_BEH</name>
            <entity>PRAM</entity>
            <body>
                …
            </body>
        </architecture>
    </vhdl_programm>
```

Представление в графической форме



Представление в виде описания на vhdl может быть импортировано в ECAD для дальнейшей работы [89,135]. Представление в виде xml используется как промежуточный уровень между описанием на языке проектирования и шаблоном созданного проектного решения. Оно используется системой и может быть использовано сторонними приложениями с целью преобразования и дальнейшего применения. Использование графической формы проектного

решения необходимо для наглядного представления как создаваемого устройства в целом, так и отдельных компонентов в частности.

В данном примере продемонстрировано применение MVC к одному элементу СФЛМ. В зависимости от количества СФЛМ, полученных после запроса к ПССЗ и получения данных из модели через контроллер, описанный вывод будет иметь несколько отличий. В графической форме будет представлена схема в виде совокупности связанных между собой элементов, а не один элемент. В xml появятся такие конструкции как <portmap>, описывающие соединения между входами и выходами элементов.

3.2. Анализ средств, применяемых в ходе получения проектных решений

Перед тем как перейти к рассмотрению методов синтеза и анализа проектных решений, необходимо рассмотреть средства, которые применяются для их получения. Особенности применения часто используемых средств в системах типа ECAD [109, 121, 153] представлены в табл.3.2.

Рассмотрим существующие и предлагаемые средства с нескольких позиций.

1. Синтез проектных решений при коллективной работе. В ходе распределенного проектирования возникают задачи объединения полученных от разных проектировщиков проектных решений. Данный показатель описывает, как именно рассматриваемые средства могут объединить проектное решение.

2. Анализ работоспособности. Данный показатель характеризует наличие ошибок в проектных решениях, полученных путем применения рассматриваемых средств.

3. Ограничения при использовании. Данный показатель характеризует наличие факторов, ограничивающих применения рассматриваемых средств.

4. Повышение качества, сокращение времени получения проектного решения. Данный показатель описывает возможности средств получения проектных решений.

5. Основные временные затраты. Данный показатель описывает самые затратные операции, выполняемые системой или проектировщиком при использовании рассматриваемых средств.

Таблица 3.2. Особенности применения средств проектирования

Показатель	**Описание на HDL языке**	**Графическое описание**	**Шаблоны типовых конструкций**	**Библиотеки элементов (IP-блоки)**	**СФЛМ**
Синтез проектных решений при коллективной работе	Система контроля версий	Система контроля версий	Конструктор, применяющий встроенные в систему проектирования структуры устройств, которые необходимо загрузить пользователю	Генератор кода, полностью готового для применения модуля. Библиотеку необходимо загрузить пользователю [156]	Система контроля версий, агент синтеза проектных решений
Анализ работоспособности	Да	Да	Да	Не требуется	Не требуется
Ограничения при использовании	Нет	Да	Количество шаблонов, сформированных пользователем	Количество элементов, сформированных пользователем	Количество СФЛМ, пополняемых с каждым проектным решением
Повышение качества, сокращение	Подсветка кода, автоматич	Отрицательное, для сложных	Отсутствие необходимости	Создание структуры	Частичная передача

времени получения проектного решения	еское форматирование, управлением комментариями, структурирование кода на группы, автоматическое завершение ключевых слов	проектов с большими схемами, огромным количество соединений	формировать типовые конструкции, а только заполнять их параметрами	из готовых наработок	работы с СФЛМ – агентам, формирование нового решения путем его конструирования из частей СФЛМ с корректировкой параметров шаблона данной модели.
Необходимость совмещения работы с другим средством проектирования	Нет	Да	Да	Нет	Нет
Основные временные затраты	Написание кода (зависимость от сложности проектируемого устройства)	Формирование схемы (зависимость от сложности схемы проектируемого устройства)	Настройка параметров шаблона под требования задачи.	Поиск необходимых элементов (зависимость от объема базы элементов)	Поиск в базе данных

Проанализировав применяемые средства, можно выделить основные продуктивные сценарии совместного использования нескольких средств получения проектных решений в ходе распределенного проектирования:

1) Создание проектных решений без использования прецедентов. Данный сценарий предполагает отсутствие проектных решений при проектировании устройства;

2) Создание проектных решений, используя только прецеденты. Данный сценарий не предполагает изменение прецедентов. Новое проектное решение получается путем использования одного или нескольких проектных решений, путем объединения выходов одних компонентов и соединения их с входами других компонентов;

3) Создание проектных решений, используя прецеденты, с редактированием параметров. Данный сценарий позволяет не только находить проектные решения, необходимые для решения задачи, но и изменять их, используя дополнительные средства проектирования.

В качестве прецедентов используются СФЛМ, IP-блоки и типовые конструкции. Описанные сценарии расположены в порядке убывания времени проектирования и увеличения скорости разработки.

3.3. Описание процесса коллективной реализации проектируемого устройства

Основную роль в процессе декомпозиции играют принципы модульности и иерархичности, т.к. они лежат в основе организации коллективной параллельной разработки устройств. Применение модульного проектирования способствует организации коллективной разработки путем разделения задач между отдельными участниками проекта, учитывая их квалификацию. Модель проектируемого устройства на языке VHDL представляет собой иерархическую структуру взаимосвязанных компонентов (Structural Architectural Body, SAB). SAB является наиболее эффективным средством создания иерархических проектов. Применение концепции SAB облегчает совместную работу нескольких

исполнителей над одним проектом [75]. Упрощается включение в новые проекты ранее созданных модулей.

Перед тем как описать применяемый метод декомпозиции, опишем процесс формирования спецификации на проектируемое устройство. Основными атрибутами создаваемой спецификации являются: название устройства, символьный код, родительский элемент, описание портов, описание команд, описание работы устройства. Результатом формирования является дерево спецификаций, представляющее собой совокупность функциональных блоков проектируемого устройства в иерархическом виде. Процесс декомпозиции организуется двумя способами: на основе задач-прецедентов и путем формирования операторов в ПССЗ на основе параметров, заложенных в спецификации.

Рассмотрим первый метод. После создания спецификации проектируемого устройства выполняется поиск задач-прецедентов в архиве ПССЗ. Атрибутами поиска являются параметры проектируемого устройства, указанного в спецификации. Поиск организуется следующим образом. Формулируется поисковый запрос путем анализа данных спецификации. Совокупность параметров спецификации с учетом оценки важности образует поисковое предписание, соответствующее запросу. Оно сопоставляется с поисковыми элементами (проектными задачами), в результате чего определяется их релевантность. Если поисковый элемент и предписание релевантные, то из поискового элемента извлекается идентификатор оператора, выдаваемого пользователю. Результатом поискового запроса является множество операторов, содержащих проектные задачи, соответствующих отобранным в процессе поиска идентификаторам.

Сценарии поиска и создания операторов ПССЗ

Для описания механизма структурного поиска задач-прецедентов в БД введем следующие обозначения: O – множество хранимых операторов ПССЗ в БД, $op_i \in O$, где op_i – i-ый оператор ПССЗ. Каждый оператор в своей структуре содержит одну или несколько проектных задач $t_j \in op_i$, где $j = 1..m_i$, m_i –

количество проектных задач в *i*-ом операторе. Введем оценки параметров спецификации $S = (S_1,S_2,\ldots,S_k)$, $k > 0$. Пусть оценка каждого параметра S_f выражается действительным числом, принадлежащим некоторому интервалу. Для определенности примем, что чем больше значение S_f, тем важнее для пользователя данный параметр. Поисковый запрос рассматривается как виртуальная задача *vt*, которая содержит параметры спецификации. Введем функцию сравнения двух задач $compFunc(t_1, t_2)$. При сравнении в идеальном случае $compFunc(vt, t)=1$, т.е. задача *t* точно соответствует виртуальной задаче *vt*.

Используя введенные обозначения, определим следующие поисковые задачи и их результаты.

1. Выполнить поиск $(op_j \in O) \mid compFunc\ (vt, t_i \in op_j) \to min$. Если результаты поиска $(op_j \in O) = 0$, то в *O* отсутствуют операторы, релевантные поисковому запросу. При $(op_j \in O) = 1$ есть единственный подходящий оператор. Если $(op_j \in O) > 1$, то таких операторов несколько.

2. Выполнить поиск $(op_j \in O) \mid compFunc\ (vt, t_i \in op_j) \le \Delta$, где Δ — оценка наибольшего допустимого расхождения смыслов поискового запроса и искомых операторов.

3. Выполнить поиск $(op_j \in O) \mid compFunc\ (vt, t_i \in op_j) \le \Delta \;\&\&\; S_f(op_j) \to \max$. Результатом поиска является подмножество операторов ПССЗ, которым соответствует наибольшая оценка важности *f*-го параметра.

Поисковые результаты выводятся с сортировкой по релевантности. В случае, когда поисковый запрос выдал результаты, проектировщик может ими воспользоваться с целью автоматической декомпозиции проектного устройства, описанного спецификацией. Данные результаты могут помочь проектировщику получить декомпозированные проектные задачи путем редактирования определенных параметров задач-прецедентов, или он может воспользоваться результатами в качестве справки.

Рассмотрим второй метод. В случае, когда поисковый запрос не выдал результатов, на основе спецификации автоматически формируется или

дополняется оператор ПССЗ. Алгоритм перехода от спецификации к оператору ПССЗ имеет вид.

1. Анализ созданной спецификации и формирование проектной задачи. Переход к шагу 2.

2. Поиск оператора, описывающего родительский элемент. При наличии оператора - переход к шагу 3, а при его отсутствии – к шагу 6.

3. Поиск оператора для интеграции проектной задачи, учитывая возможность использования проектного решения, полученного при решении данной задачи, в качестве независимого компонента. При наличии оператора - переход к шагу 6, а при его отсутствии – к шагу 4.

4. Подсчет количества дочерних операторов в родительском элементе. Если количество операторов больше 0, то переход к шагу 5. В противном случае – к шагу 7.

5. Изменение существующего типа родительского элемента на оператор декомпозиции. Создание оператора синтеза и установка с ним связи всех дочерних элементов первого уровня вложенности. Переход к шагу 6.

6. Поиск проектировщика для решения проектной задачи. Переход к шагу 7.

7. Добавление / редактирование оператора в БД ПССЗ.

Назначение проектной задачи проектировщику

В представленном алгоритме перехода от спецификации к оператору ПССЗ был определен этап поиска проектировщика для решения проектной задачи. Описываемые алгоритмы лежат в основе «балансировщика нагрузки» на проектировщика. При наличии найденных задач-прецедентов, выполняется следующий алгоритм:

1) Выбирается время, затраченное на выполнение задач из списка. Выполняется переход к шагу 2;

2) Выбирается проектировщик и формируется ассоциативный массив, где ключом является идентификатор проектировщика, а значением – время, затраченное на выполнение задачи. Выполняется переход к шагу 3;

3) Сортируется массив по времени выполнения. Выполняется переход к шагу 4;

4) Для каждого проектировщика проверяется текущая загруженность, путем анализа его очереди задач и времени их выполнения. Определяется и запоминается крайний срок начала выполнения рассматриваемой задачи путем поиска в очереди задач, между которыми можно разместить новую задачу, учитывая приоритет выполнения. Выполняется переход к шагу 5;

5) Определяется проектировщик, который менее загружен и способен реализовать проектную задачу. Выполняется переход к шагу 6;

6) Пересчет значения крайнего срока у всех задач из очереди после добавленной задачи. Выполняется переход к шагу 7;

7) К рассматриваемой задаче прикрепляются проектные решения путем их связи через идентификаторы. Это необходимо для того, чтобы проектировщик при создании проектного решения улучшил качество и сократил время, затрачиваемое на реализацию путем использования решения в готовом виде или с редактированием определенных параметров. Выполняется переход к шагу 8;

8) Сохранение данных проектной задачи. Выполняется переход к шагу 9;

9) Конец алгоритма.

При отсутствии найденных задач-прецедентов выполняется алгоритм поиска проектировщиков, способных решить рассматриваемую задачу:

1) Выполняется поиск проектировщика в системе. Если результат поиска > 0 выполняется переход к шагу 2, в противном случае – к шагу 8;

2) Для каждого проектировщика проверяется текущая загруженность путем анализа его очереди задач и времени их выполнения. Определяется и запоминается крайний срок начала выполнения рассматриваемой задачи путем поиска в очереди задач, между которыми можно разместить новую задачу, учитывая приоритет выполнения. Выполняется переход к шагу 3;

3) Определяется проектировщик, который менее загружен и способен реализовать проектную задачу. Выполняется переход к шагу 4;

4) Пересчет значения крайнего срока у всех задач из очереди после добавленной задачи. Выполняется переход к шагу 5;

5) Сохранение данных проектной задачи. Выполняется переход к шагу 6;

6) Конец алгоритма.

Для описания механизма структурного поиска проектировщиков в БД введем следующие обозначения: D – множество проектировщиков в системе, данные которых хранятся в БД, $d_i \in D$, где d_i – i-ый проектировщик. Данные о каждом проектировщике, содержат информацию:

1) стаж работы $exp \in d_i$;

2) знания в области цифровой схемотехники $know_j \in d_i$, где j = 1..m, m – количество областей схемотехники, по которым предприятие, на котором работает проектировщик, реализует проектные решения;

3) знание языков программирования $lang_b \in d_i$, где b = 1..n, n – количество языков программирования, которые применяются на предприятии с целью реализации проектных решений;

4) навыки при работе с программами типа ECAD $skills_t \in d_i$, где t = 1..v, v – количество программам типа ECAD, используемых на предприятии с целью реализации проектных решений.

Информация о i-ом проектировщике представляет собой $info_i = (exp, know, lang, skills)$.

Введем оценки параметров, соответствующих информации о проектировщике $S = (S_1,S_2,\ldots,S_k)$, $k>0$. Пусть оценка каждого параметра S_f выражается действительным числом, принадлежащим некоторому интервалу. Для определенности примем, что чем больше значение S_f, тем важнее данный параметр. Поисковый запрос рассматриваться как объявление *ads*, которое содержит требуемую информацию о проектировщике. Введем функцию сравнения информации о проектировщике с объявлением *compDevFunc(ads, info)*. При сравнении, в идеальном случае $compDevFunc(ads, info_i)=1$, т.е. информация $info_i$, о проектировщике d_i, точно соответствует поданному объявлению *ads*.

Используя введенные обозначения, определим следующие поисковые задачи и их результаты:

1) Выполнить поиск $(d_j \in D) \mid compDevFunc\ (ads, info_i \in d_j) \to min$. Если $d_j = 0$, то в D отсутствуют проектировщики, релевантные поисковому запросу. При $| d_j | = 1$, есть единственный подходящий проектировщик. Если же $| d_j | > 1$, то таких проектировщиков несколько;

2) Выполнить поиск $(d_j \in D) \mid compDevFunc\ (ads, info_i \in d_j) \leq \Delta$, где Δ — оценка наибольшего допустимого расхождения смыслов поискового запроса (объявления) и данных о искомых проектировщиках;

3) Выполнить поиск $(d_j \in D) \mid compDevFunc\ (ads, info_i \in d_j) \leq \Delta \;\&\&\; S_f(d_j) \to max$. Результатом поиска является множество проектировщиков, которым приписана наибольшая оценка важности f-го параметра;

Поисковые результаты выводятся с сортировкой по релевантности. Результатом описанных алгоритмов является повышение качества управления процессом коллективного проектирования достигаемое путем сокращения времени на формирование типовых задач.

3.4. Поиск проектных решений СФЛМ в распределенной среде проектирования

Одной из основных функций системы распределенного проектирования является обеспечение эффективного поиска СФЛМ. В условиях распределенной архитектуры самым простым вариантом является поиск среди всех баз данных (плоская модель поиска) [126]. Такой подход может быть эффективным при сравнительно небольшом количестве баз данных. В противном случае возникает ситуация, при которой поиск производится в базах данных, не соответствующих запросу пользователя. В результате релевантность найденной информации резко снижается. В дополнение к этому возникают проблемы производительности при «широковещательном» поисковом запросе. Решение проблемы может заключаться в ограничении множества баз данных, в которых производится поиск информации. Наиболее логичным вариантом выделения областей поиска является тематическая фильтрация, которая предполагает, что запрос

пользователя снабжается метками принадлежности к той или иной предметной области цифровой схемотехники. Обладая полной информацией о привязке баз данных к тематическому классификатору, осуществляется выделение тематических областей цифровой схемотехники и предоставляется эта информация компоненту, выполняющему непосредственный запрос на поиск СФЛМ. При этом сохраняется возможность гибкого поиска по базам данных путем задания принадлежности к любому уровню классификатора, что позволяет искать сразу в нескольких тематических областях цифровой схемотехники.

Для организации поиска используют фактографические системы, которые дают точные и полные ответы на запрос пользователя, но это становится возможным только при весьма серьезных ограничениях и на представление информации в таких системах, и на язык запросов.

Также используются документальные системы. В таких системах в ответ на свой запрос пользователь получает документы или некоторые их части.

У большинства систем проектирования для хранения библиотечных элементов используется файловый способ хранения. При этом отдельный файл соответствует одному элементу библиотеки. Отсутствие средств автоматизированного поиска библиотечных элементов по критериальным параметрам в существующих системах увеличивает время проектирования.

Учитывая фундаментальный принцип создания распределенных баз данных, для пользователя распределенная система должна выглядеть так же, как нераспределенная система [147]. Будет рассмотрен поиск к базе данных, а распределенность достигается с помощью брокеров [106], решающих задачу маршрутизации запросов пользователей. Подобная задача состоит в последовательном выполнении следующих действий:

1) Привязка (binding) к серверу. Результатом является адрес машины, на которую нужно передать вызов;

2) Маршалинг (marshaling). Результатом является упаковка вызова процедуры и ее аргументы в сообщение в некотором формате;

3) Сериализация (serialization). Результатом является преобразование в поток байтов полученного сообщения и отправка с помощью какого-либо протокола, транспортного или более высокого уровня;

4) Десериализация (deserialization). Результатом является преобразование потока байтов в сообщение;

5) Демаршалинг (unmarshaling). Результатом является распаковка вызова процедуры и ее аргументов из принятого сообщения;

6) Вызов локально соответствующей функции поиска серверного компонента и получение поискового результата.

3.5. Синтез проектных решений в процессе распределенного проектирования

Для выполнения синтеза проектных решений в описанных выше сценариях сформулированы следующие способы [150].

Прямое распределенное проектирование

В данном методе процесс создания проектного решения в виде кода на VHDL заключается в работе команды проектировщиков через СРП, позволяющую управлять версиями проекта. Как пример реализации рассматривается взаимодействие с системой git. Данный метод может быть применен при разработке программной модели несложной схемы или внесении изменения в существующий исходный код для расширения функциональных возможностей модели [62].

Разработка программной модели осуществляется в текстовом редакторе с поддержкой того языка проектирования, в котором ведется описание. Поддержка обычно осуществляется в виде подсветки синтаксиса языка, оперативного выведения ссылок помощи на дальнейшее правила написания операторов [63]. Эта модель синтеза подходит для автоматизации процесса написания VHDL кода командой проектировщиков. Структура системы, в основу которой положен метод прямого распределенного проектирования, изображена на рис. 3.2.

На рисунках основной ветвью разработки в системе git является Trunk; дополнительной ветвью разработки, в которой вводится или проверяется

некоторая функциональность, является Branch; операцией слияния различных ветвей проекта является Merge [26,15];

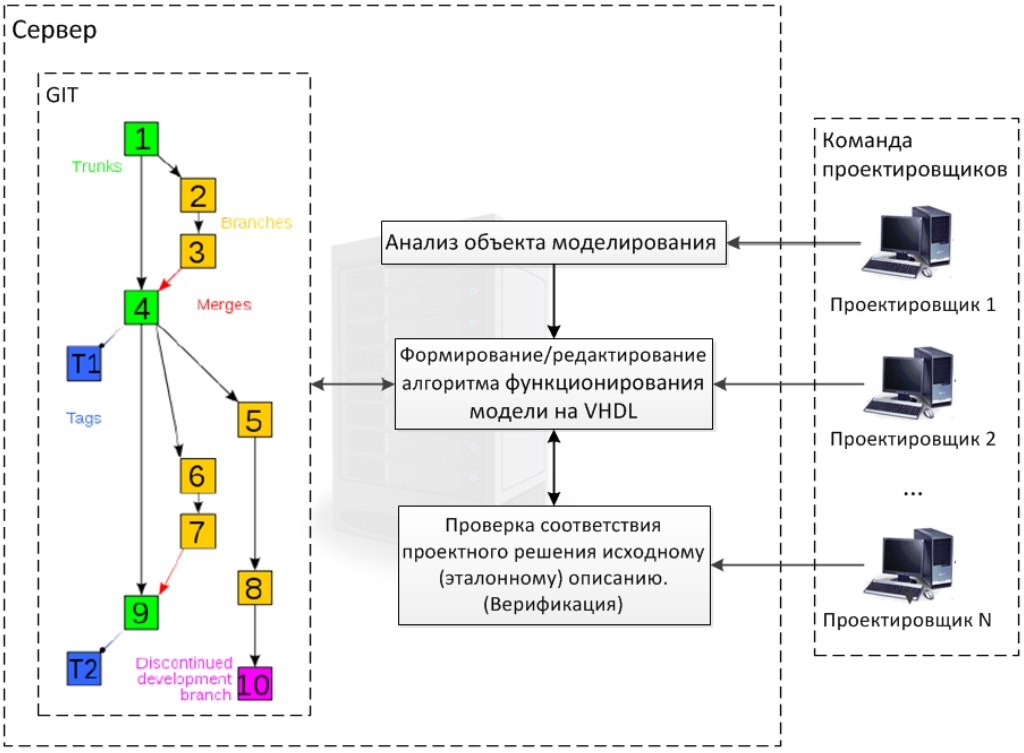


Рис. 3.2. Структура системы, в основу которой положен метод прямого распределенного проектирования

Распределенное проектирование с подстановкой

Следующим этапом автоматизации процесса синтеза программ написанных на языке типа VHDL является внедрение подстановки. Особенностями этого способа синтеза является формирование СФЛМ из исходного кода программы с последующим сохранением его в базу данных. Процесс подстановки заключается в интеграции уже написанных шаблонов СФЛМ с частичным изменением параметров модели в создаваемое проектное решение.

После анализа объекта моделирования проектировщик переходит к разработке СФЛМ. Новое проектное решение может быть получено в результате синтеза созданных ранее СФЛМ, которые могут быть найдены в базе данных. После создания СФЛМ проводится предварительное тестирование с целью создания условия для испытания путем измерения критериальных параметров и оптимизации их значений. В этом методе вводится элемент поиска СФЛМ для сокращения времени

варьирования структуры и параметров. Этап испытаний и оптимизации позволяет произвести окончательное тестирование разрабатываемой модели. Внесение изменений в СФЛМ происходит в том случае, если параметры не удовлетворяют требованиям. Структурная и параметрическая модификация требует изменений исходного кода, проектировщиками.

Это решение масштабируется на случай взаимодействия от 1 до L серверов и от 1 до M команд проектировщиков. Связь между серверами осуществляется с целью поиска проектных решений, которые представлены в виде СФЛМ. Команды проектировщиков могут работать на разных предприятиях и серверах, которые соединены в одну большую сеть.

Структура системы, в основу которой положен метод распределенного проектирования с подстановкой, изображена на рис. 3.3.

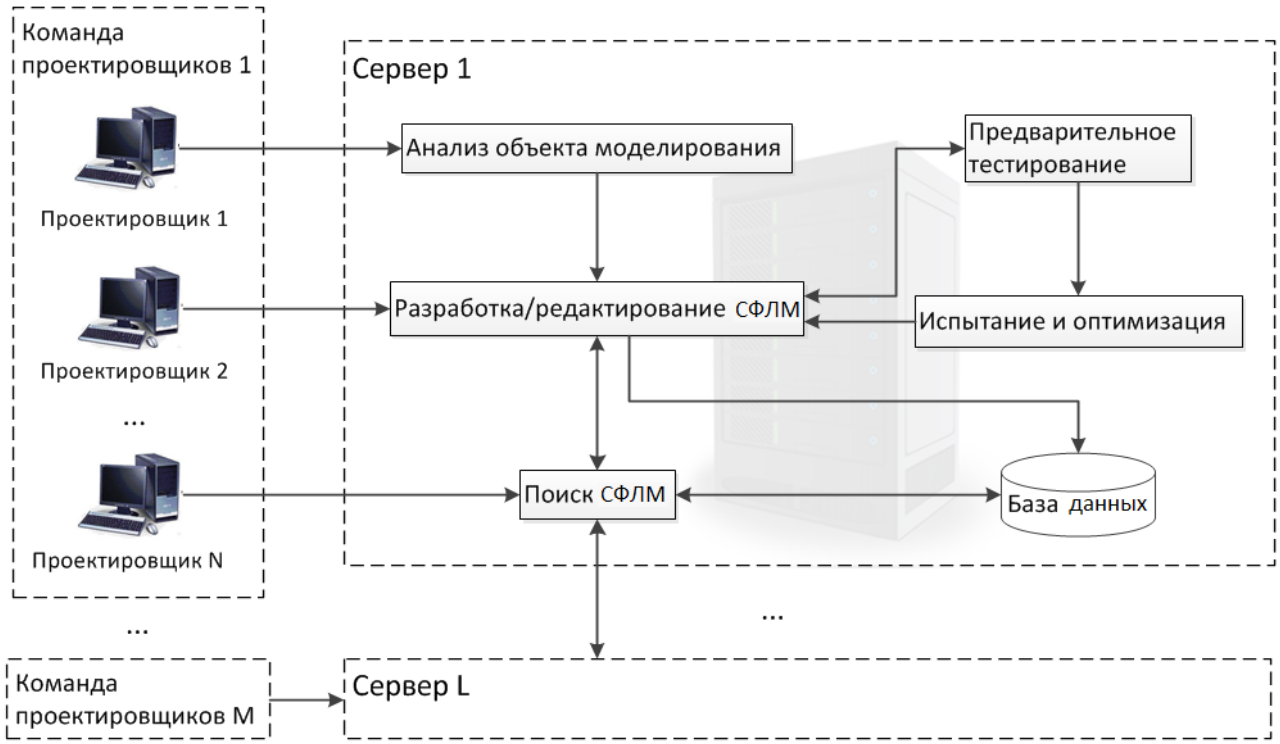


Рис. 3.3. Структура системы, в основу которой положен метод распределенного проектирования с подстановкой.

Автоматическая генерация СФЛМ по шаблону

Еще одним из этапов автоматизации процесса синтеза программ написанных на языке типа VHDL является применение метода автоматической генерации СФЛМ по шаблону. В отличии от СФЛМ, который был описан в

предыдущей части статьи, шаблон не содержит в описании объектов и связей между объектами прямого назначения размерности переменных. Входные переменные шаблона позволяют рассчитать размерность ячеек памяти, отвечающих за структурные элементы объектов и связей между ними.

В базе данных хранятся некоторое множество шаблонов типовых конструкций функциональных моделей на языке HDL, из которых можно создавать новое проектное решение путем оперирования параметрами шаблона, не вникая в тонкости используемого языка. Для описания недостающих элементов проектировщику необходимо внести изменение в сам шаблон, но для этого необходимо знание языка HDL.

Таким образом, автоматическая генерация СФЛМ является полностью автоматизированной, но ограничена количеством шаблонов сохраненных в системе проектирования. Для работы с шаблонами необходимо воспользоваться мастером, который поможет получить проектное решение путем генерирования кода из шаблона с подставленными параметрами. Дальнейшие действия по созданию или редактированию программ на VHDL ложатся на проектировщика. В систему введен git с целью синтеза кода, полученного из шаблонов, проектировщиками.

Структура системы, в основу которой положен метод автоматической генерации СФЛМ по шаблону, изображена на рис. 3.4.

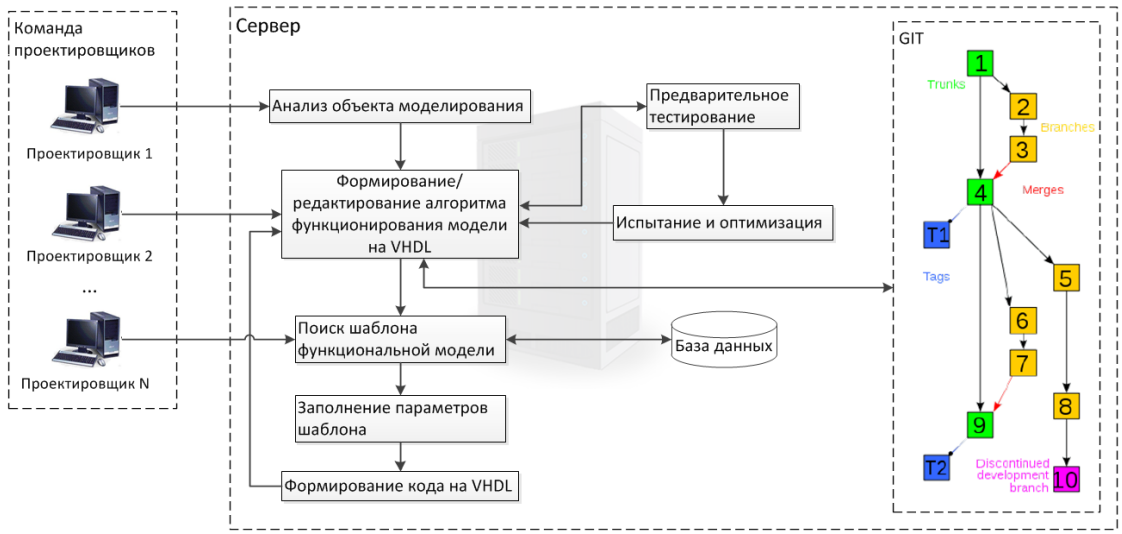


Рис. 3.4. Структура системы, в основу которой положен метод автоматической генерации СФЛМ по шаблону.

Автоматическая генерация СФЛМ с подстановкой

Заключительным является применение метода автоматической генерации СФЛМ с подстановкой. Автоматическая генерация СФЛМ с подстановкой является сочетанием второго и третьего метода синтеза проектного решения. Особенностью этого метода является то, что можно не только подобрать необходимую СФЛМ из сохраненных в базе, но и внести изменения в алгоритм функционирования с последующим созданием шаблона СФЛМ и сохранением в базу данных. В этом методе СФЛМ представляет собой шаблон с набором входных параметров, заполнив которых можно получить новую СФЛМ.

Это решение масштабируется на случай взаимодействия от 1 до L серверов и от 1 до M команд проектировщиков. Связь между серверами осуществляется с целью поиска проектных решений, которые представлены в виде СФЛМ. Команды проектировщиков могут работать на разных предприятиях и серверах, которые соединены в одну большую сеть.

Структура системы, в основу которой положен метод распределенного проектирования с подстановкой, изображена на рис. 3.5.

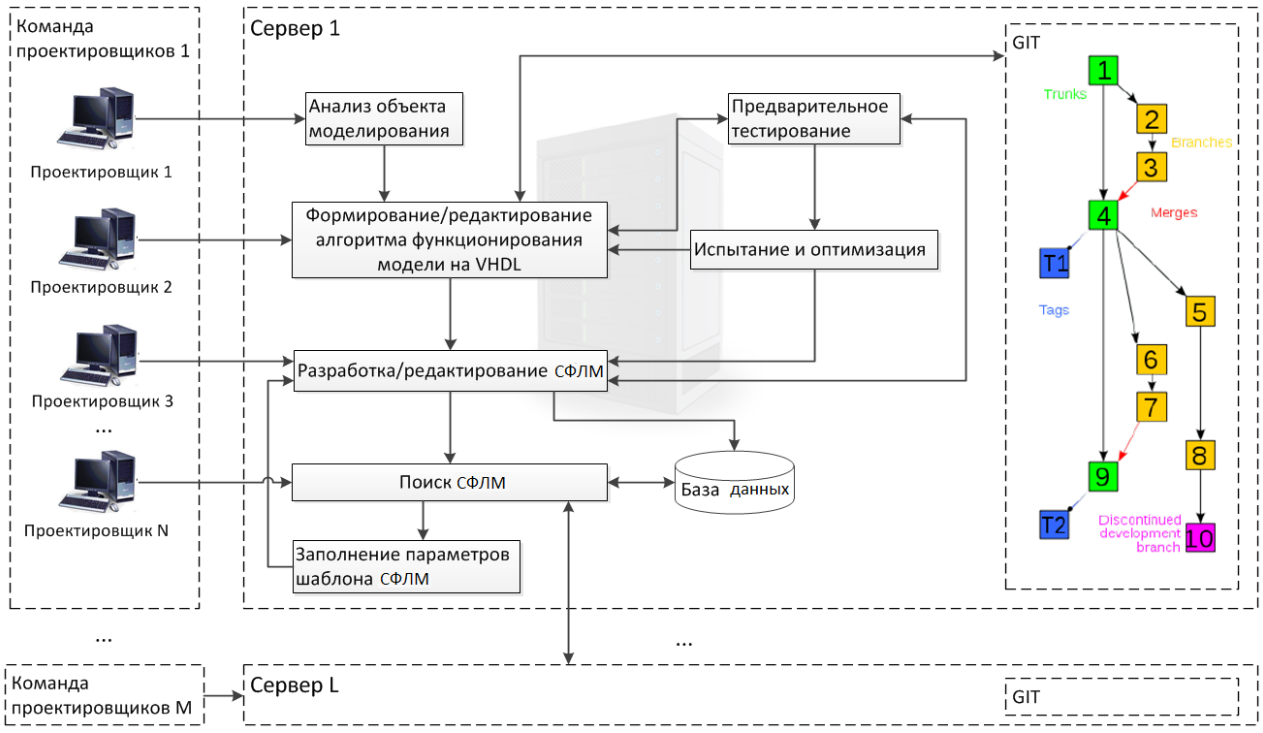


Рис. 3.5. Структура системы, в основу которой положен метод автоматической генерации СФЛМ с подстановкой.

3.6. Расчет коэффициента повторяемости кода при реализации HDL-проектов на основе СФЛМ

Процесс написания reusable (многократно используемого) кода очень важен. Использование готового кода, который является работоспособным и был хорошо протестирован, оказывается экономически более выгодным в 90% случаев для компаний производителей. В случае использования готового модуля, сторонний разработчик может вообще не знать о внутреннем устройстве, при этом ему важен лишь интерфейс и принцип работы. Проектировщик может посмотреть на спецификацию модуля, и используя простой интерфейс к нему (например, задав пару параметров, отвечающих за чувствительность) уже может включить этот модуль в свой проект.

Из основных требований к написанию reusable кода выделяются следующие:

1) документирование и написание комментариев;

2) Модуль должен быть максимально параметризирован;

3) Написание многократно используемых процедур и функций;

4) Написание отдельных testbench для каждого разрабатываемого модуля.

Более полное и детальное описание требований по проектированию цифровых устройств на VHDL для ПЛИС представлено в [92].

Рассмотрим два подхода к расчету коэффициента повторяемости кода.

Первый подход основан на возможности применения СФЛМ, хранящихся в базе проектных решений, путем поиска в ней по параметрам подходящих СФЛМ. Так для микропрограммного устройства управления (МПУУ) на основе ассоциативной памяти, состоящего из 20 блоков, реализованного на VHDL, при проектировании выделены две группы элементов, разработанных на основе повторного использования кода.

К первой группе таких элементов относятся регистры признаков ($Рг_4$, $Рг_8$, $Рг_{10}$), регистры индикации ($Рг_{11}$, $Рг_{13}$) и регистр маски ($Рг_{20}$).

Ко второй группе таких элементов относится память признаков (M_3), память признаков безусловных ассоциаций (M_7), память признаков условных ассоциаций (M_9)

Коэффициент повторного использования VHDL-кода функционального блока проектируемого устройства вычисляется по следующей формуле: $K_i = \frac{U\_line_i}{N\_line_i} \cdot 100\%$, где U_line_i - количество строк в i-ом функциональном блоке, используемые при построении нового функционального блока, N_line_i - общее количество строк в i-ом функциональном блоке, $i = 0 .. N$, где N – общее количество функциональных блоков в проектируемом устройстве.

Средний коэффициент повторного использования равен $K_{повт.исп.} = \frac{\sum K_i}{N}$

Данный коэффициент основывается на факте, что большинство функциональных блоков могут быть получены из уже существующих путем частичной модификации кода. При анализе проекта МПУУ для первой группы коэффициент повторного использования равен 51%, для второй группы коэффициент повторного использования равен 28%. Средний коэффициент повторного использования равен $K_{повт.исп.} = \frac{51 + 28}{2} = 39{,}5\%$.

В качестве второго примера рассмотрены несколько функциональных модулей процессора, такие как АЛУ (ALU), блок регистров (Register block), блок счетчика команд (PC) и т.д. Проанализируем код функциональных блоков процессора [45], посчитаем коэффициенты повторного использования при реализации блоков в процессорах [42, 19] и результаты занесем в табл. 3.3.

Таблица 3.3. Коэффициенты повторного использования для модулей процессора

Функциональный модуль	**CPU [42]**	**CPU [19]**
1. ALU	27%	29%
2. Register block	18%	20%
3. PC	26%	24%

Средний коэффициент повторного использования равен 24%.

Проанализируем код модулей ALU, RAM и ROM из различных источников [30,32,39,40,41,44] и посчитаем коэффициент повторного использования:

1) ALU – 32%;

2) RAM – 38%;

3) ROM – 22%.

Второй подход основан на применении декларации generic для описания параметризованных моделей. Данная декларация позволяет создавать модели различной структуры и поведения. В декларации задаются значения параметров, которые характеризуют связь с внешним окружением. Примерами таких параметров являются температура, количество выводов компонента схемы, временные задержки и т.п. Данный подход требует от проектировщика написание и тестирование более сложной логики работы модуля, которая будет учитывать параметры декларации generic.

Параметры generic могут быть использованы в качестве параметров СФЛМ, и, таким образом, служить в качестве поисковых при анализе базы проектных решений.

Коэффициент полноты использования логики работы i-ого функционального блока при j-ом наборе параметров generic вычислим по следующей формуле:

$K_{ij} = \frac{U\_line_{ij}}{N\_line_i} \cdot 100\%$, где U_line_{ij} - количество используемых строк кода в i-ом функциональном блоке при j-ом наборе параметров generic, N_line_i - количество строк кода в i-ом функциональном блоке. $i = 1 .. N$, где N – общее количество функциональных блоков в проектируемом устройстве, $j = 1 .. M$, где M – количество вариантов использования i-ого функционального блока в проектируемом устройстве при различном наборе параметров generic.

Учитывая, что в проектируемом устройстве может быть несколько раз использован модуль с различными параметрами, необходимо использовать следующую формулу определения коэффициента полноты использования

логики работы для функционального блока: средний коэффициент полноты использования логики работы i-ого функционального блока – $K_{полн.исп.i} = \frac{\sum K_{ij}}{M}$

.

Тогда средний коэффициент полноты использования определяется так: $K_{полн.исп} = \frac{\sum K_{полн.исп.i}}{N}$.

В частности, данная декларация используется для проекта калькулятора [10]. Коэффициент полноты использования для данного функционального блока равен 67%. Рассмотренный подход на основе декларации generic является параметрическим приемом проектирования VHDL-программ, относящимся к повторному использованию кода [28]. Выведенный коэффициент, его значение для примера, а также возможность использования параметров generic в качестве параметров СФЛМ говорит о позитивных преимуществах предлагаемых в диссертации методах и средствах.

Практическая важность повторного использования HDL-кода отмечается также в работе [13]. По данным фирмы FLIR, являющейся ведущим производителем передовых тепловизионных систем, включая тепловизоры, камеры воздушного слежения и системы машинного зрения, с использованием MATLAB HDL Coder [17] коэффициент повторяемости составил 30% [13]. Продукты данной фирмы играют ключевую роль в промышленных, коммерческих, государственных областях более чем в 60 странах мира**.** Являясь основоположником индустрии производства тепловизионных камер, компания поставляет тепловизионное и оборудование ночного видения в научные, промышленные, государственные и военные организации уже более 50 лет. В областях диагностического обслуживания, мониторинга состояния объектов, недеструктивного тестирования, НИОКР, медицинских разработках, измерения температуры и тестирования для органов правопорядка, наблюдения и безопасности, контроля процесса производства, FLIR предлагает самый широкий выбор тепловизоров для начинающих и профессионалов.

Увеличение повторяемости с 0 до 30% и появление общего хранилища алгоритмов, привело к сокращению сроков проектирования.

3.7. Оценка эффективности применения СФЛМ, как средства получения проектного решения

Процесс получения проектного решения СФЛМ может быть представлено как дискретный процесс (*S*), состоящий из множества этапов (*n*) и проектных задач (m): $S = \{S_i\}$, где $i = \overline{1,n}$.

Каждое подмножество S_i в свою очередь – это множество проектных задач *i* этапа проектирования: $S = \{S_i^j\}$, где $i = \overline{1,n}$, $j = \overline{1,m}$.

Одним из основных количественных критериев, является время разработки проектного решения T_p.

Время реализации проектной задачи *i* на этапе *j* определяется по формуле

$$T_p = \sum_{i=1}^{n} T_i = \sum_{i=1}^{n} \sum_{j=1}^{m} T_i^j .$$

При учете времени реализации особое внимание уделяется факторам процесса проектирования, таким как сложность задачи, скорость разработки, тестирования и т.д. Рассмотрим метрику для оценки технических характеристик и факторов разработки.

Существуют множество решений, позволяющих оценивать трудозатраты по разработке программного [21, 154]. Оценка по количеству строк кода является одной из самых распространенных. Оно было использовано в качестве основы методики COCOMO (Constructive Cost Model) [85]. В данной методике оценивается количество строк кода (Lines Of Code - LOC). Основная формула в методике COCOMO имеет следующий вид:

$$E = a \times KSLOC^b \times EAF, \qquad (3.1)$$

где в зависимости от проекта вычисляются параметры *a* и *b* путем регрессионного анализа; *E* – затраты в человеко-месяцах; *KSLOC* – количество строк кода (в тыс. и без учета комментариев); для корректировки трудозатрат в формуле используется параметр EAF (Effort Adjustment Factor) [85].

Варьирование фактора *EAF*, в зависимости от состояния среды, позволяет увеличить или уменьшить трудозатраты:

$$EAF = \prod_{i=0}^{n} Ci \text{, (3.2)}$$

где C_i – фактор среды.

При проектировании ВУ на языке описания аппаратуры, оценка трудозатрат производится по формулам (3.1) и (3.2). В ходе анализа будут отдельно оценены каждые функциональные требования, а константу *b* примем равной 1. Учитывая зависимость объема от вида работы, а время реализации от мощности ресурса, которое используется при выполнении. Преобразуем формулу (3.1) к виду:

$E = \sum_i \sum_j (\frac{KSLOC(i, j)}{Wj} \times EAF)$, где *j*-ый вид работы характеризуется его мощностью *Wj* и количеством строк кода *i*-го требования *KSLOC(i,j)*.

Проводится анализ аналогичного по архитектуре и языку проектирования разрабатываемого модуля и определяются параметры *KSLOC* и *W*. Выделяются три типа программного кода.

1. Код, описывающий интерфейс системы. Обычно это код, описывающий сам интерфейс. Для проектируемых компонентов на языке VHDL – это блок entity. Обозначим его как SLOCR.

2. Код, обрабатывающий элементы интерфейса и связывающий его с другими частями системы. Для проектируемых компонентов на языке VHDL – это испытательные стенды для проверки правильности работы созданной модели проектируемого устройства и пакеты проекта, представляющие собой описание констант, подтипов, часто используемых процедур. Обозначим его как SLOCI.

3. Код, обеспечивающий определенную логику работы системы. Для проектируемых компонентов на языке VHDL – это блок architecture. Обозначим его как SLOCP.

Используя следующую формулу, рассчитываем KSLOC:

$$KSLOC = \frac{(SLOCP \times Kp + SLOCI \times Ki + SLOCR \times Kr)}{1000},$$

где *Kp* – фактор изменения логики работы системы; *Ki* – фактор изменения обработки интерфейсных элементов системы; *Kr* – фактор изменения интерфейса системы; Факторы *K* подбираются опытным путем. Например, для изменений менее 20% фактор изменений может быть равен 0,2; для 50% изменений – 0,7; для вновь создаваемых компонент – 1,0.

Учитывая разработку предыдущих компонентов, опытным путем определяется мощность W.

Опишем фактор учета сложности и опыта разработки (*ФУСОР*), который определяется эмпирически и состоит из следующих значений:

1) вносятся корректировки в уже существующий код = 1,00;

2) компонент разрабатывался, есть опыт = 1,10;

3) есть опыт и легко можно внести изменения = 1,30;

4) опыта нет, но решения может быть найдено с помощью сторонней помощи = 2,00;

5) отсутствует опыт и помощь = 4,00.

Описанный выше фактор зависит от конкретной личности или группы.

Длительность проектирования и тестирования вычисляется по формуле:

$$T = \sum_i \sum_j \left(\frac{KSLOC(i, j)}{Wj} \times ФУСОР(i)\right) \times \frac{1}{P}.$$

Количественными метриками программы, такие как среднее число строк для проектируемого устройства. При анализе учитываются как объективные (длина, скорость разработки, скорость тестирования и т.д.) так и субъективные (простота и понятность кода).

По описанной выше методике получим количественные характеристики проектирования на языке VHDL. В качестве типовых решений возьмем процессор RISC CPU [129] и микропрограммное устройство управления [62].

Данные по характеристикам SLOC для типовых проектов представлены в табл. 3.4.

Таблица 3.4. Данные по характеристикам SLOC для типовых проектов

Описание работы	**SLOC**		
	Объем кода логики SLOCP	**Объем кода обработки SLOCI**	**Объем кода SLOCR**
Средний модуль процессора RISC CPU	390	90	35
Средний модуль микропрограммного устройства управления MICRO	217	15	10

Будем оценивать проектирование устройств на языке описания аппаратуры с применением СФЛМ и без нее. Результирующие оценки занесем в табл. 3.5. Для оценки эффективности рассмотрим:

1) пример модели микропрограммного устройства управления, состоящего из 19 функциональных блоков: 7 регистров, 4 блока сравнения, 3 блоков памяти, схемы формирования управляющего сигнала, дешифратора, анализатора многократного совпадения, схемы формирования сигнала записи и блока управления [62];

2) Пример модели процессора RISC, состоящего из 11 функциональных блоков: пакет проекта, арифметико-логическое устройство, блок регистров, блок счетчика команд, блок управления прерываниями, ядро процессора, память программ, память данных, порт ввода-вывода, модель варианта процессора, испытательный стенд [129].

Мощность выполнения работ при создании подобных систем составляет W= 2,5. Длительность работы будем вычислять исходя из расчета работы трех проектировщиков.

При прямом проектировании:

$$SLOC_{micro} = \sum_{l=1}^{17}(217\times1+15\times1+10\times1) = 4114\,[\text{строк}],$$

$$SLOC_{cpu} = \sum_{l=1}^{11}(390 \times 1 + 90 \times 1 + 35 \times 1) = 5665\,[\text{строк}],$$

$$KSLOC_{micro} = 4{,}114\,[\text{строк}],$$

$$KSLOC_{cpu} = 5{,}665\,[\text{строк}].$$

Фактор учета сложности и опыта примем равным 4 (Нет ни опыта, ни помощи).

$$T_{micro} = \frac{4{,}114}{2{,}5 \times 3} \times 4 = 2{,}194\ [\text{мес}],$$

$$T_{cpu} = \frac{5{,}665}{2{,}5 \times 3} \times 4 = 3{,}021\ [\text{мес}].$$

При проектировании с применением шаблонов:

$$SLOC_{micro} = \sum_{l=1}^{17}(217 \times 0{,}6 + 15 \times 0{,}5 + 10 \times 0{,}55) = 2434{,}4\,[\text{строк}],$$

$$SLOC_{cpu} = \sum_{l=1}^{11}(390 \times 0{,}6 + 90 \times 0{,}5 + 35 \times 0{,}55) = 3280{,}75\,[\text{строк}],$$

$$KSLOC_{micro} = 2{,}434\,[\text{строк}],$$

$$KSLOC_{cpu} = 3{,}281\,[\text{строк}].$$

Фактор учета сложности и опыта примем равным 2 (Опыта нет, но есть помощь).

$$T_{micro} = \frac{2{,}434}{2{,}5 \times 3} \times 2 = 0{,}649\ [\text{мес}],$$

$$T_{cpu} = \frac{3{,}281}{2{,}5 \times 3} \times 2 = 0{,}875\ [\text{мес}].$$

При проектировании с использованием библиотечных элементов:

$$SLOC_{micro} = \sum_{l=1}^{17}(217 \times 0{,}2 + 15 \times 0{,}3 + 10 \times 0{,}3) = 865{,}3\,[\text{строк}],$$

$$SLOC_{cpu} = \sum_{l=1}^{11}(390 \times 0{,}2 + 90 \times 0{,}3 + 35 \times 0{,}3) = 1270{,}5\,[\text{строк}],$$

$$KSLOC_{micro} = 0{,}865\,[\text{строк}],$$

$$KSLOC_{cpu} = 1{,}271\,[\text{строк}].$$

Фактор учета сложности и опыта примем равным 1 (Надо поправить то, что есть).

$$T_{micro} = \frac{0{,}865}{2{,}5 \times 3} \times 1 = 0{,}115 \text{ [мес]},$$

$$T_{cpu} = \frac{1{,}271}{2{,}5 \times 3} \times 1 = 0{,}169 \text{ [мес]}.$$

При проектировании с использованием СФЛМ:

$$SLOC_{micro} = \sum_{l=1}^{17} (217 \times 0{,}19 + 15 \times 0{,}2 + 10 \times 0{,}2) = 785{,}91 [\text{строк}],$$

$$SLOC_{cpu} = \sum_{l=1}^{11} (390 \times 0{,}19 + 90 \times 0{,}2 + 35 \times 0{,}2) = 1090{,}1 [\text{строк}],$$

$$KSLOC_{micro} = 0{,}786 \text{ [строк]},$$

$$KSLOC_{cpu} = 1{,}090 \text{ [строк]}.$$

Фактор учета сложности и опыта примем равным 1 (Надо поправить то, что есть).

$$T_{micro} = \frac{0{,}786}{2{,}5 \times 3} \times 1 = 0{,}105 \text{ [мес]},$$

$$T_{cpu} = \frac{1{,}090}{2{,}5 \times 3} \times 1 = 0{,}145 \text{ [мес]}.$$

Таблица 3.5. Результаты по SLOC метрике

	Прямое проектирование	**Проектирование с применением шаблонов**	**Проектирование с использованием библиотечных элементов**	**Проектирование с использованием СФЛМ**
RISC CPU	2,194	0,875	0,115	0,105
MICRO	3,021	0,649	0,169	0,145

Применение СФЛМ в качестве библиотечного элемента способствует сокращению времени разработки и повышению качества проектных решений за счет минимизации ошибок в коде.

Данная методика оценки трудозатрат может быть использована в практической деятельности по проектированию ВУ на языке типа HDL, при

анализе текущего состояния проекта, прогнозировании сроков его окончания и стоимости. В отличие от других методик она, в известной степени, обладает свойством самообучения. Это позволяет, с одной стороны, достаточно легко выбирать базовые значения для новых проектных задач, не имеющих аналогов реализации, а, с другой стороны, – накапливать и использовать опыт предыдущей разработки.

3.8. Выводы

1. Представлен процесс работы с СФЛМ по концепции MVC, являющейся схемой построения архитектурного каркаса, которая часто используется при проектировании технических объектов.

2. Описан процесс коллективной разработки проектируемого устройства путем его декомпозиции на более простые компоненты, с последующим формированием проектных задач.

3. Представлены алгоритмы поиска задач-прецедентов, проектировщиков и распределение задач между проектировщиками. Данные алгоритмы направлены на повышение качества управления процессом коллективного проектирования достигаемое путем сокращения времени на формирование типовых задач.

4. Описан поиск проектных решений СФЛМ в распределенной среде проектирования, способствующий организации процесса распределенного проектирования, а также уменьшению нагрузки на систему, в которой был выполнен запрос.

5. Описаны способы синтеза проектных решений в процессе распределенного проектирования, которые способствуют повышению уровню автоматизации с целью снижения времени разработки, требований к квалификации проектировщиков и организации процесса взаимодействия проектировщиков, находящихся на удаленном расстоянии друг от друга.

6. Проведен анализ повторности использования кода путем ввода коэффициента повторяемости. В результате был получен коэффициент равный 25-30%, характеризующий сокращение времени проектирования при повторном

использовании СФЛМ. Также был проведен анализ полноты использования кода в проектах, применяющих декларацию generic с целью создания параметризованных модулей и их многократного применения. Возможность использования параметров generic в качестве параметров СФЛМ говорит о позитивных преимуществах предлагаемых в диссертации методах и средствах.

7. Применение SLOC метрик позволяет изучить сложность проекта, оценить объем работ, стилистику разрабатываемой программы и время, затраченное на проектирование. Представленная оценка эффективности применения СФЛМ при поиске и формировании нового проектного решения позволяет получить количественную характеристику, обозначающую преимущество рассматриваемого подхода, путем сокращения времени проектирования в среднем на 10%.

ГЛАВА 4. РЕАЛИЗАЦИЯ МНОГОАГЕНТНОЙ СИСТЕМЫ РАСПРЕДЕЛЕННОГО ПРОЕКТИРОВАНИЯ

В данной главе перейдем к вопросу о реализации экспериментально-практической части диссертационной работы, а именно, построению многоагентной СРП.

Представим в виде UML диаграмм общее представление функционального назначения системы и поведения, наиболее важных частей системы на основе указания последовательности передаваемых сообщений. Опишем структуру базы данных.

Описание будем проводить в порядке появления основных элементов во второй и третьей главе.

4.1. Разработка web-ориентированного транслятора VHDL описаний в шаблоны структурно-функциональных лингвистических моделей

Одной из технологий работы системы является использование частного облака, как средства организации процесса распределенного проектирования с централизованным хранением проектных решений. Под частным облаком будем понимать инфраструктуру, предназначенную для использования одной организацией, включающей несколько подразделений [16].

В качестве платформы взаимодействия была выбрана технология CORBA, созданная для поддержки разработки и развертывания сложных объектно-ориентированных прикладных систем. CORBA является механизмом в программном обеспечении для осуществления интеграции изолированных систем, который дает возможность программам, написанным на разных языках программирования, работающих в разных узлах сети, взаимодействовать друг с другом так же просто, как если бы они находились в адресном пространстве одного процесса.

Серверная часть системы основана на шаблоне MVC (модель-представление-контроллер), который позволяет разделить систему на три легко модифицируемых и практически независимых компонента. Каждый из этих компонент отвечает за различные задачи в системе [9].

Управление запросами пользователя, имеющие типы GET и POST, осуществляется через **контроллер**. Основной функцией является вызов и координация действий, необходимых ресурсов и объектов. Контроллер вызывает модель для системной задачи и выбирает подходящее представление для ее отображения. Контроллер получает запросы от проектировщиков и обеспечивает соответствующую реакцию, характеризующуюся контролем входных данных, получением модели и рендер представления сформированных данных.

Модель представляет собой совокупность правил функционирования системы. Она, реагируя на запросы, предоставляет данные (проектные решения, задачи и т.д.) и методы для работы с данными, изменяет свое состояние.

Представление обеспечивает различные способы отображения данных, которые получены из модели. В системе присутствуют представления в виде шаблонов, которые заполняются данными. Представление формирует вывод данных в различных форматах, таких как html, json, xml и т.д., в зависимости от обращения в системе проектирования.

Процесс функционирования системы по шаблону MVC представлен на рис. 4.1.

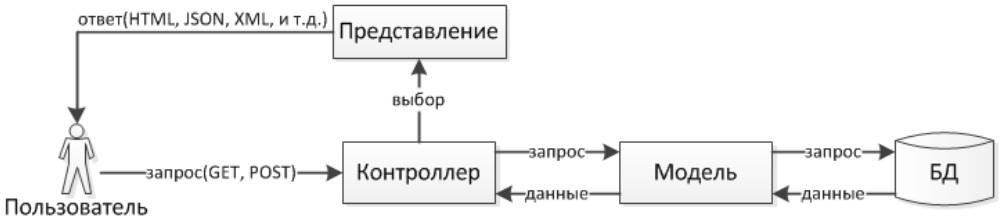


Рис. 4.1. Процесс функционирования системы по шаблону MVC

Демонстрация шаблона MVC на операциях авторизации и анализа VHDL кода представлена в виде диаграммы последовательности [38], изображенной на рис. 4.2.

В основе разработанной СРП лежит методология нисходящего проектирования: от спецификаций на части проектируемого устройства до формирования проектного решения, описывающего функционирование устройства.

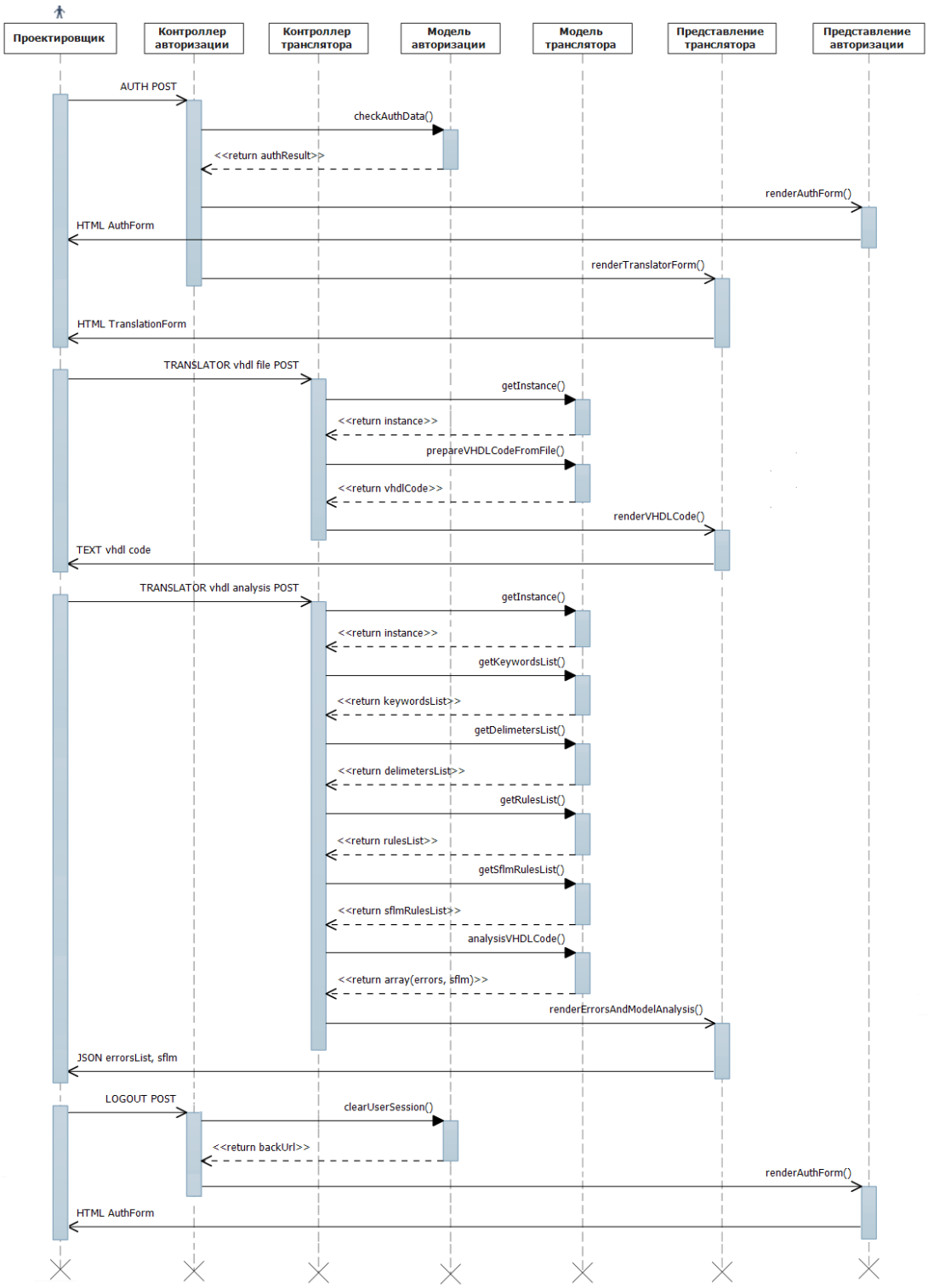


Рис. 4.2. Диаграмма последовательности операций авторизации и анализа VHDL кода, основанных на шаблоне MVC

Программы на языке VHDL могут состоять из одного или более файлов. Пара объект – архитектура, является неотъемлемой частью каждого файла программы. Под объектом понимается интерфейсная часть, указывающая на то, как включать данный объект внутри другого объекта, находящегося на более

высоком уровне иерархии, а в архитектуре описан алгоритм его функционирования. Одному объекту может соответствовать несколько архитектур, описывающих разные алгоритмы функционирования. Таким образом, учитывая структуру программы на языке VHDL, можно создавать библиотеки объектов, процедур, типов и т.п., выполнять решение проектных задач в ходе проектирования проекта параллельно несколькими проектировщиками, использовать объекты из других проектов, тестировать проекты по одинаковой методике.

В рассматриваемой системе возникает задача управления потоками проектных работ. Для этого предложена ассоциативно-ориентированная модель управления задачами (работами) в виде ПССЗ, которая детально описана в главе 2 параграф 2.4, позволяющая эффективно организовать коллективное проектирование VHDL-объектов.

Рассмотрим программную реализацию СРП. Процесс функционирования СРП сложных VHDL-объектов с экранными формами, демонстрирующими этот процесс, показан на рис. 4.3. Цифрами представлен путь проектирования от формирования спецификации до заполнения базы данных проектными решениями [1]. (A) – форма демонстрирующая формирование VHDL кода памяти программ (PRAM) в web-редакторе с подсветкой синтаксиса. (B) – схематичное отображение формирования VHDL кода памяти данных (DRAM) в web-редакторе с подсветкой синтаксиса. (C) – форма демонстрирующая формирование редактирование спецификации памяти программ (PRAM) [129].

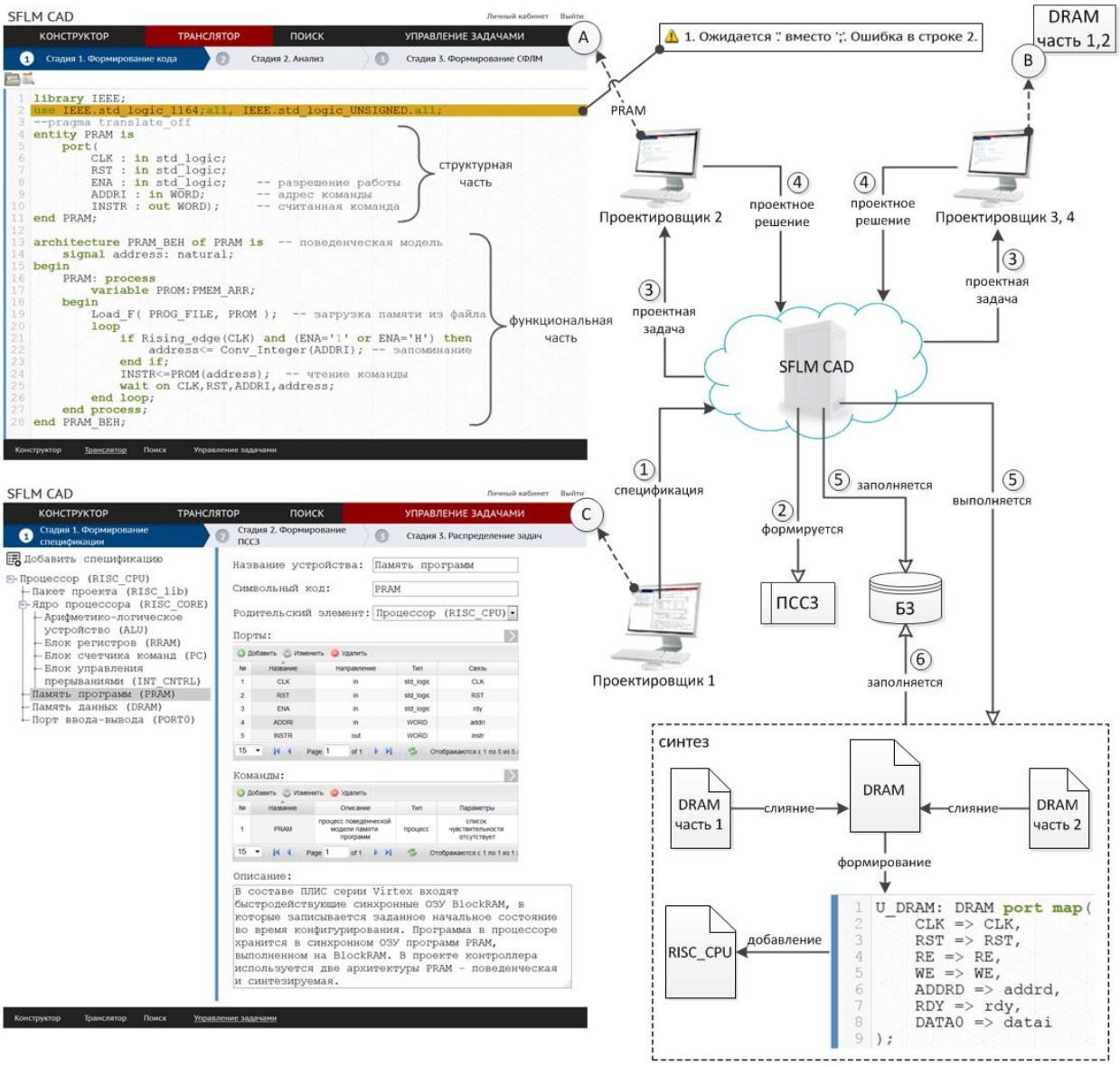


Рис. 4.3. Процесс функционирования системы распределенного проектирования сложных VHDL-объектов

Основными операциями синтеза являются слияние частей проектируемого устройства. Слияние выполняется на различных стадиях разработки:

1) процесс слияния кода VHDL программ производится через объединение веток разработки в распределенной системе управления версиями GIT;

2) процесс слияния СФЛМ производится через объединение блоков в полноценное проектное решение, решающее поставленную проектную задачу;

3) указание связи между компонентами путем формирования карты портов на основе спецификации и ее интеграция в родительский элемент.

Сформированная СФЛМ позволяет организовать удобный поиск проектных решений по различным критериям, таким как тип проектируемого

устройства, наличие портов и др. Учитывая, что СФЛМ представляет собой совокупность функциональных блоков проектируемого устройства, то процесс создания нового проектного решения может представлять собой как написание нового кода, так и конструирование из различных блоков.

Принцип повторного использования шаблонов СФЛМ является основополагающим в рассматриваемой СРП. Т.е. СФЛМ созданные одними проектировщиками могут быть использованы другими для получения нового проектного решения, которое также сохраняется в базу данных.

Структура описываемой web-ориентированной системы хранится в виде PHP-скриптов и отдельных файлов, называемых шаблонами. Для хранения контента системы, данных ПССЗ и проектных решений используется база данных. Агенты, описанные выше, в процессе своей работы обрабатывают данные, полученные из базы данных в виде проектных решений или операторов ПССЗ. Реализация web-интерфейса позволило минимизировать требования к аппаратному и программному обеспечению рабочих мест пользователей, устранить привязку к определенной операционной системе(ОС) и использовать открытые ОС(Linux), т.к. разработанная система является кроссплатформенной.

4.2. Программное и информационное обеспечение

СРП рассматривается как объект, который может одновременно играть роль и клиента, и сервера. Если объект является инициатором вызова метода у другого объекта, то он выполняет роль клиента. Если объект является исполнителем запрашиваемого метода, то он выполняет роль сервера. Большинство объектов одновременно исполняют роль и клиентов и серверов, попеременно вызывая и исполняя методы. Применение технологии CORBA способствует созданию более гибких систем, чем клиент-сервер, которые основаны на двухуровневой и трехуровневой архитектуре.

Применение web-технологий, таких, как php, html, javascript позволяет сформировать удобный web-интерфейс. Посредством этого интерфейса проектировщики могут работать в одной системе, управляя проектными задачами и создавая проектные решения через браузер. В качестве серверного

программного обеспечения были выбраны популярный web-сервер – Apache. СУБД – MySQL. Прокси сервер – Nginx.

Такой выбор программного обеспечения обусловлен необходимостью повышения производительности системы, т.е. для снижения нагрузки, мощный и популярный веб сервер apache слишком требователен к ресурсам. Поэтому обычно статичную информацию предоставляет прокси сервер nginx, а динамичную информацию (серверные скрипты) обрабатывает apache.

Информационное обеспечение

Информационное обеспечение СРП включает в себя внесистемное и внутрисистемное информационное обеспечение.

В состав внесистемного информационного обеспечения входят поставщики данных, являющиеся информационными системами, которые предоставляют необходимые данные для последующей работы с ними в СРП.

В состав внутрисистемного информационного обеспечения входят следующие компоненты: интегратор, который получает и обрабатывает данные от поставщиков; хранилище данных, представляющее собой базу данных проектных решений, а также базу данных агентов в системе распределенного проектирования. Хранилище данных предназначено для хранения данных системы, проектных решений, проектных задач, данных агентов и обмена информации между агентами.

Внутрисистемное информационное обеспечение организовано по принципу создания центрального хранилища данных. Ядром системы является база данных, содержащая всю информацию о сформированных проектных решениях в виде СФЛМ. Каждое проектное решение в системе представлено отдельной сущностью в базе данных, которая имеет уникальный идентификатор.

Поиск в базе данных организуется следующим образом. Запрос формируется на ограниченном естественном языке и подвергается анализу, в рамках которого выделяются дескрипторы, присутствующие в словаре. В соответствие запросу ставится совокупность дескрипторов, которая участвует в получении релевантности путем сопоставления с поисковыми образами. Если

было установлено семантическое соответствие поискового запроса и поискового образа проектного решения, то из образа извлекается идентификатор проектного решения, которое необходимо передать проектировщику, сделавшему запрос. Ответом на запрос является множество проектных решений, соответствующих отобранным в процессе поиска идентификаторам.

Проектные решения СФЛМ в базе данных сохраняются в унифицированной форме. Это позволяет для проектных решений, представленных на различных языках описания, иметь единое представление в хранилище данных.

Хранение параметров и описаний СФЛМ в базе данных позволяет реализовать детализированный поиск необходимой модели при помощи языка запросов SQL.

4.3. Представление функционального назначения и поведения наиболее важных частей системы

Традиционно проблема управления информационными ресурсами решается на базе подхода, основанного на централизации сервисов хранения и обработки. Наряду с достоинствами данного подхода, заключающимися в простоте реализации и обслуживания системы, организации единой точки доступа ко всем ресурсам, существует ряд серьезных недостатков, сдерживающих развитие систем с такой архитектурой.

Основные из них – это необходимость в дорогостоящем аппаратно-программном обеспечении для организации обработки огромных объемов данных и поступающих запросов пользователей, жесткие управленческие схемы администрирования, отсутствие мотивации владельца передавать свои проектные решения в централизованные хранилища и др. Актуальное значение при разработке подобных систем приобретает идея децентрализации.

При реализации предлагаемого подхода отсутствует необходимость перемещения информации в какое-либо централизованное хранилище. Проектное решение размещается там, где оно было создано проектировщиком. Для этого ему предлагается web-система с простым и удобным

пользовательским интерфейсом, позволяющим создавать и использовать проектные решения в виде СФЛМ.

Распределенный подход к поиску и работе с СФЛМ, находящейся в различных базах данных, является основой предложенной архитектуры СРП. При таком подходе осуществляется непосредственная разработка и поиск СФЛМ в распределенной сети предприятия, состоящей из доступных через Интернет, территориально-распределенных локальных систем SFLM CAD, взаимодействующих между собой по протоколу GIOP/IIOP. Необходимо принять во внимание, что при данном подходе не предлагается полный отказ от реализации идеи централизованного управления, так как для обеспечения взаимодействия компонентов географически-распределенной системы [139] (локальных систем SFLM CAD) необходимо наличие управляющего звена (CLOUD SFLM CAD), обеспечивающего структуризацию множества систем, распределенный поиск, сбор статистики.

Процесс распределенного проектирования можно рассматривать с двух сторон. Во-первых, распределенность достигается за счет организации процесса решения сложной проектной задачи несколькими проектировщиками, которые могут находиться на удаленном расстоянии друг от друга. Во-вторых, система через которую проектировщики решают задачи является географически-распределенной, т.е. имеет один общий центр (управляющее звено) и несколько локальных систем установленных на серверах филиалов предприятия.

Взаимодействие компонентов географически-распределенной системы представлено в виде диаграммы использования, изображенной на рис. 4.4.

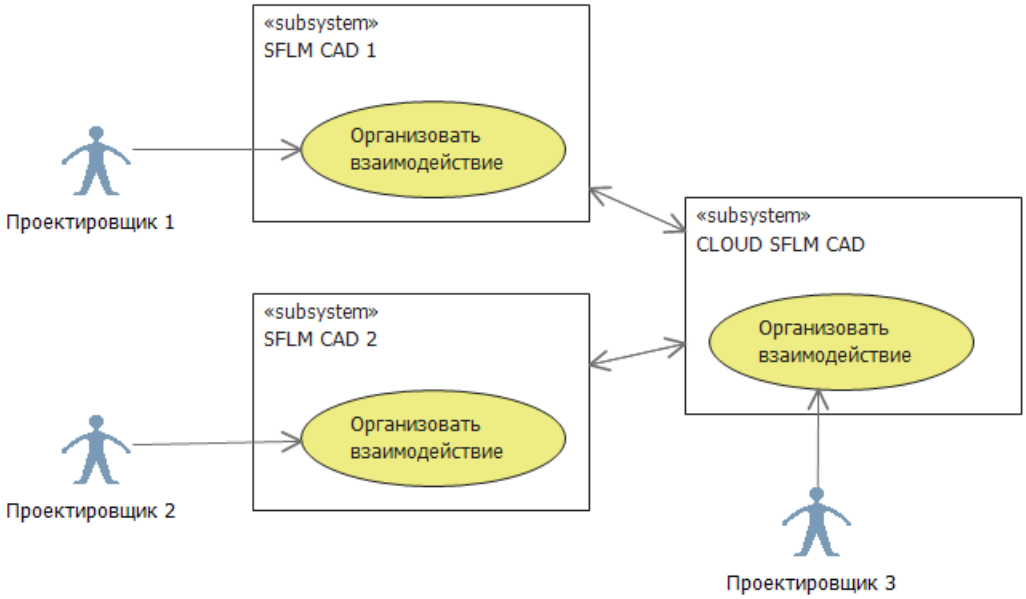


Рис. 4.4. Взаимодействие компонентов географически-распределенной системы

Взаимодействие проектировщиков и агентов в СРП представлено в виде диаграммы использования, изображенной на рис. 4.5. На данной диаграмме агенты рассматриваются как компоненты в распределенной системе [140]. Процесс взаимодействия представлен как распределенный поиск проектных решений в виде СФЛМ.

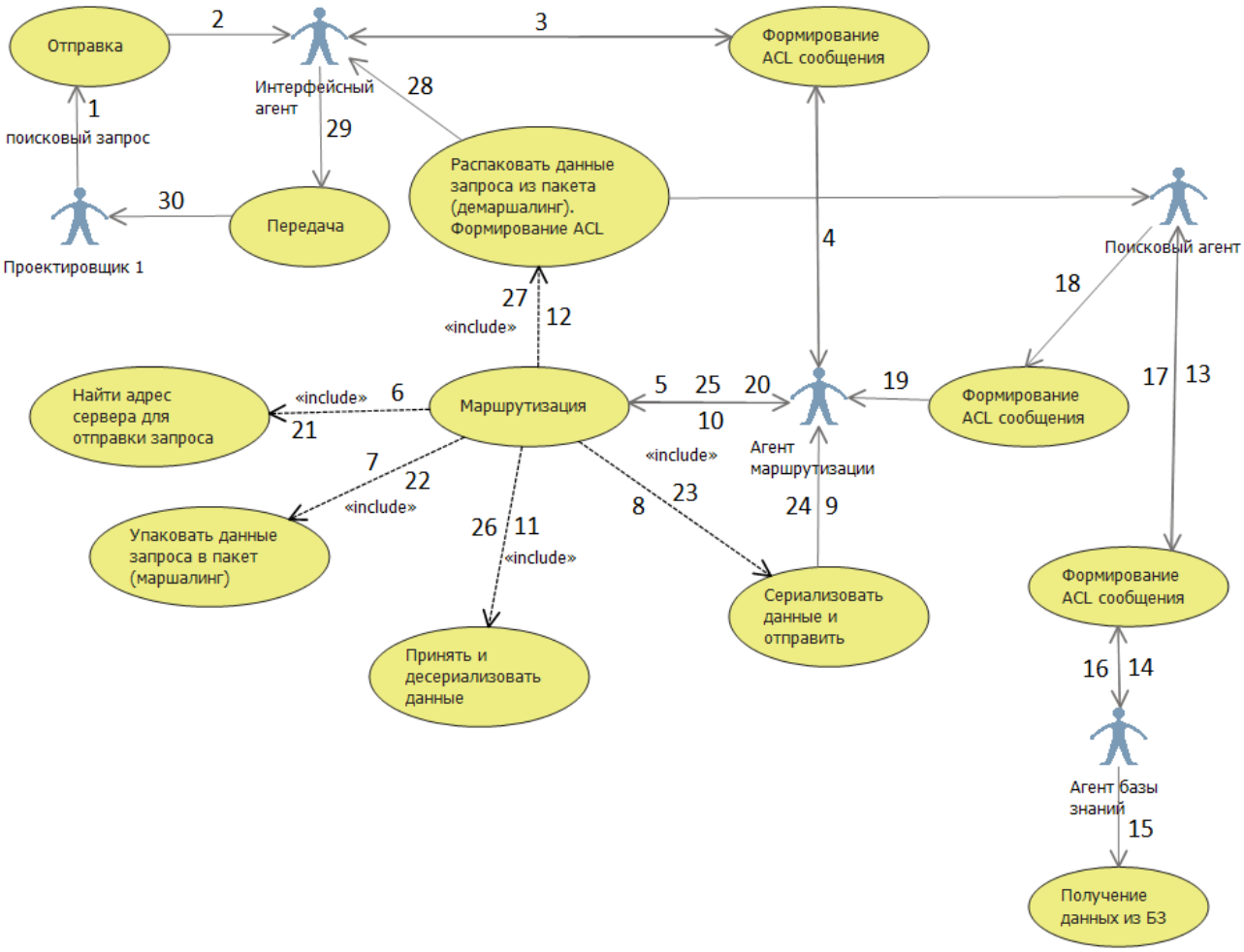


Рис. 4.5. Взаимодействие агентов в географически-распределенной системе

Взаимодействие между агентами производится посредством формирования ACL-сообщения с последующей его отправкой. ACL-сообщение имеет вид:

(inform

: sender INA

: receiver ROA

: content search_query_data

: in-reply-to prev_message_id

: ontology projectSolution

: reply-by 29.08.2014 12:00

) .

Взаимодействие между агентом и пользователем происходит путем передачи данных в метод контроллера системы.

Опишем работу web-ориентированного транслятора в виде диаграммы активности. Разработанная версия транслятора работает по следующему алгоритму (рис. 4.6).

1. При входе на страницу транслятора можно написать VHDL код в редакторе или загрузить из файла.

2. После написания кода проводятся анализ, в процессе которого формируется список ошибок.

3. Если есть ошибки, то они отображаются, в противном случае происходит формирование xml представления СФЛМ.

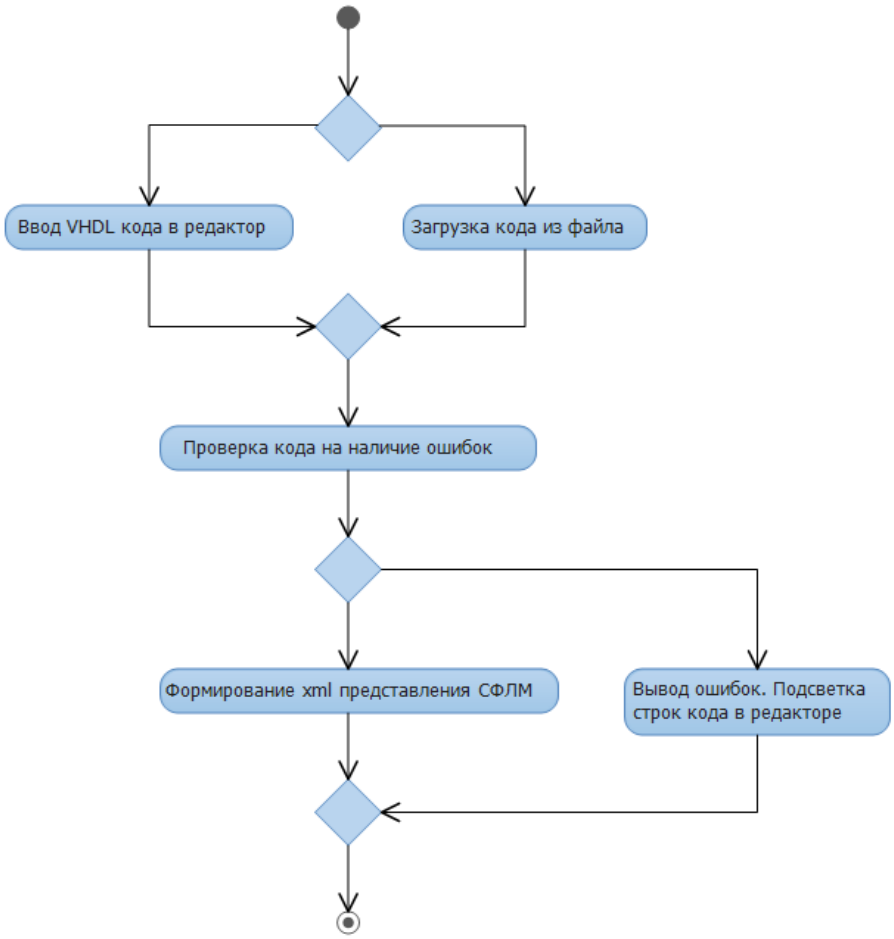


Рис. 4.6. Функционирование web-ориентированного транслятора VHDL описания в СФЛМ

Для коллективной удаленной работы предложена система на облачной архитектуре с целью взаимодействия нескольких подразделений предприятия с настройкой системы под нужды этих подразделений.

Локальная версия системы, расположенная на сервере предприятия, имеет ограниченный внешний доступ (только для компонентов географически-распределенной системы). Подобная организация дает преимущество в скорости работы, т.к. передача данных по локальной сети происходит быстрее, чем через сеть интернет. Также допускается возможность организации работы при отсутствии сети интернет.

Функциями СРП являются:

1) Поддержка иерархического тематического классификатора областей схемотехники;

2) Управление проектными задачами;

3) Ведение реестра распределенных систем с привязкой их к узлам тематического классификатора;

4) Поддержка механизма статистики. Сбор, хранение и представление информации;

5) Формирование проектных решений с использованием web-ориентированного транслятора;

6) Поддержка контроля версий разрабатываемых проектов;

7) Предоставление интерфейса пользователя с возможностью поиска информации по базам данных распределенной системы;

8) Компоненты данной информационной системы взаимодействует друг с другом посредством реализации механизма обмена сообщениями.

Описание сообщений в СРП имеет вид:

1) Запрос на регистрацию удаленной системы в качестве компонента распределенной среды;

2) Поисковый запрос к компонентам распределенной среды через центральный элемент в виде облачной системы для получения проектных решений СФЛМ, которые соответствуют заданным пользователем поисковым критериям;

3) Поисковый запрос к компонентам распределенной среды через центральный элемент в виде облачной системы для поиска задач-прецедентов и проектировщиков способных решить проектную задачу.

4.4. Представление структур хранения данных

Рассмотрим основные элементы базы данных СРП. На рис. 4.7. представлена структура БД с указанием таблиц следующих функциональных назначений:

1. необходимые для функционирования системы и взаимодействия с ней пользователей (users, loginHash, userGroup и т.д.);

2. предназначенные для работы с проектными задачами (tasks, PNSTStructure и т.д.);

3. предназначенные для работы транслятора HDL описания в СФЛМ (rules, ruleXML, keyItems);

4. предназначенные для формирования спецификаций (specifications, specificationPorts, specificationCommands);

Заполнение БД происходит непосредственно проектировщиками, в процессе решения практических задач в СРП [57]. Основными элементами, хранящимися в БД проектных решений, являются СФЛМ. Механизм поиска представляет собой процедуру поиска проектных задач в базе данных по запросу проектировщика с указанием критериальных параметров интересующего проектного решения.

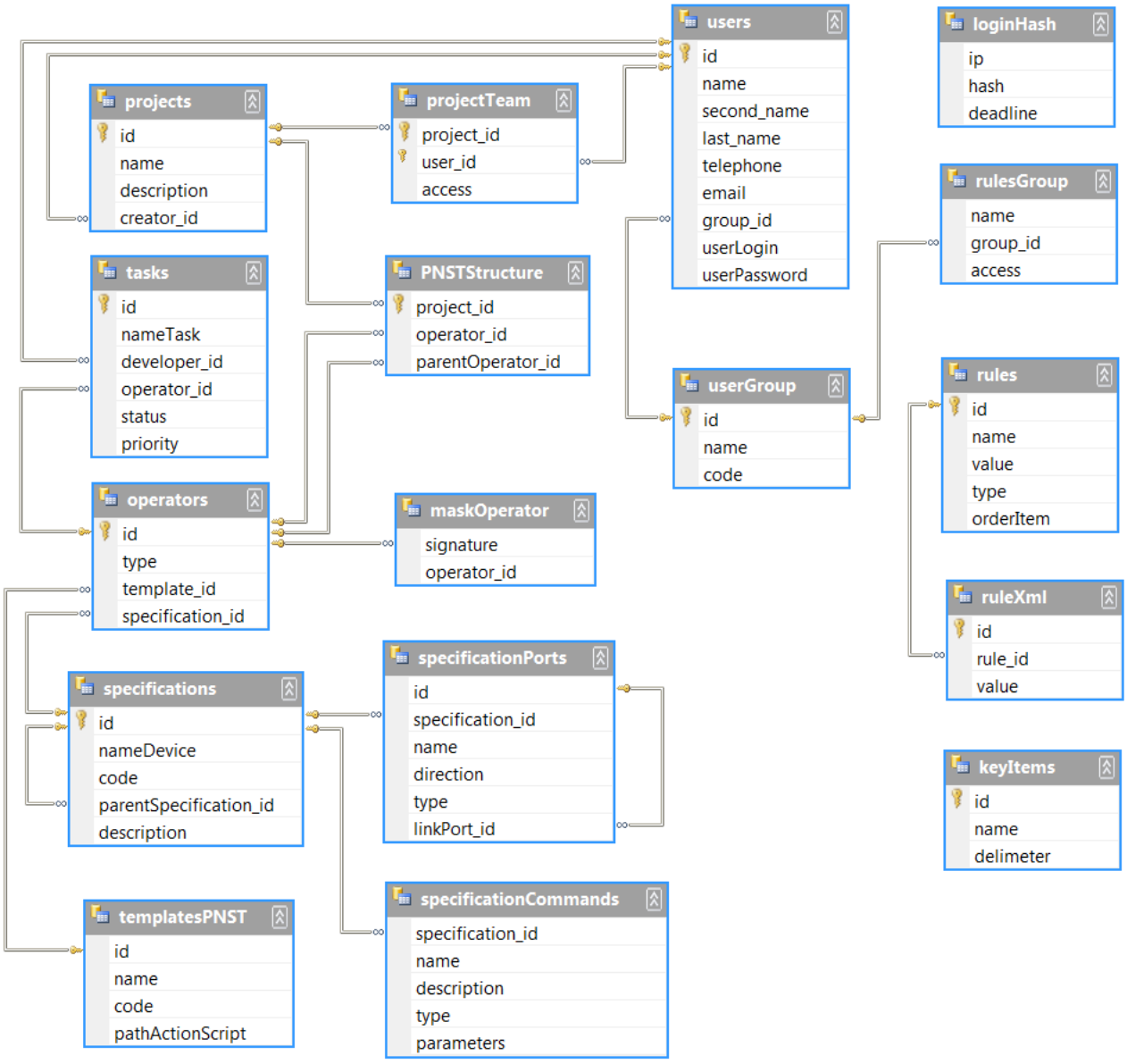


Рис. 4.7. Частичная структура БД СРП

Данные об агентах также хранятся в таблицах. Для функционирования агентов выделяются следующие таблицы, которые представлены на рис. 4.8.

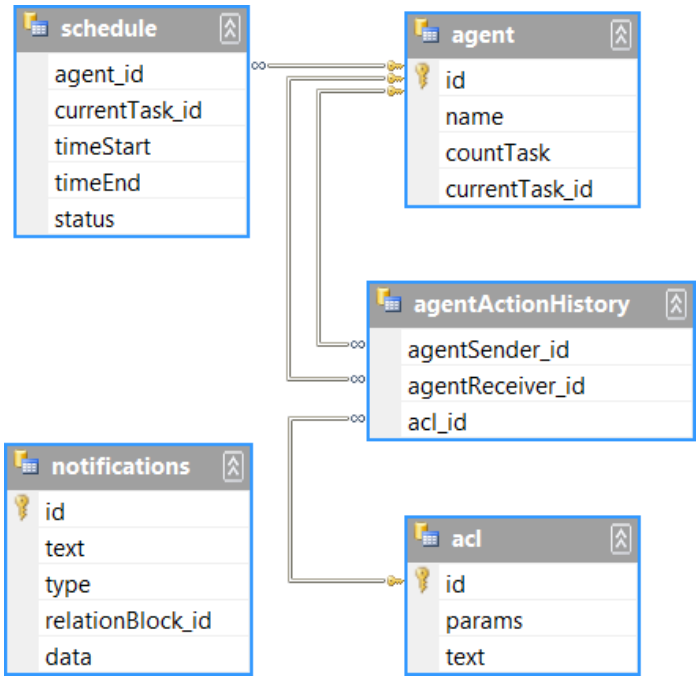


Рис. 4.8. Таблицы, содержащие данные об агентах

Данные СФЛМ распределяются между несколькими таблицами. Данное разделение было обусловлено структурой объекта СФЛМ. На рис. 4.9. представлены данные таблицы.

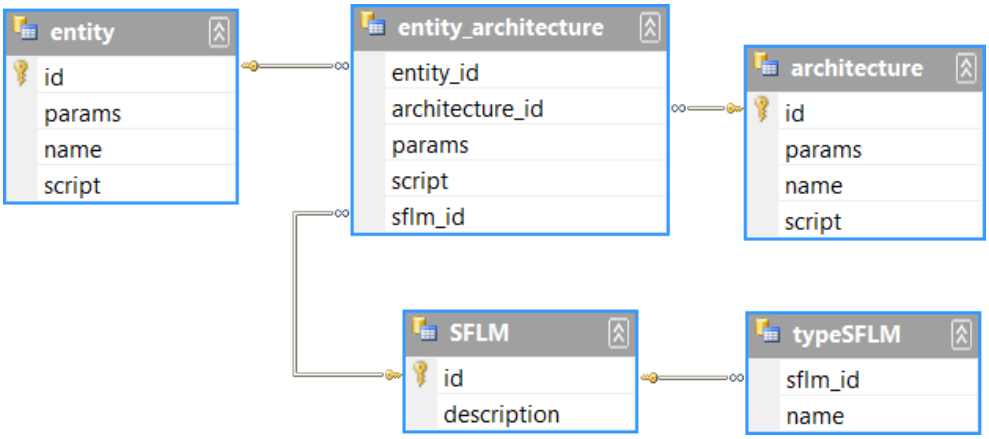


Рис. 4.9. Таблицы для данных СФЛМ

4.5. Организация процесса интеграции в системе распределенного проектирования

Рассмотрим виды интеграции в порядке увеличения сложности, интеллектуальности и требований к наличию дополнительного оборудования.

1. Интеграция со сторонними системами, путем обмена файлами в формате *.vhd. Данный вид интеграции самый простой и подходит для всех ECAD систем, которые работают с языком VHDL.

2. Интеграция со сторонними системами через репозиторий (svn, git). Данный вид интеграции позволяет осуществлять синхронизацию проектов между системами. Для подобной интеграции необходимо наличие в сторонней системе возможности загрузки из репозитория [123, 127]. При наличии специальной программы, такой как TortoiseGit, TortoiseSVN и т.д., проектировщик может получить копию проекта для дальнейшей работы с ним. Доступ проектировщика к проекту организуется через SSH-ключ, таким образом организуется защита проекта от скачивания пользователями не участвующими в проекте.

3. Интеграция со сторонними сервисами.

Подобная интеграция позволяет проектировщикам, работая через интерфейс системы, организовать поиск проектных решений и получить данные (при их наличии) со стороннего ресурса [101,144]. Это вид интеграции поможет сократить время поиска проектного решения через Internet самим проектировщиком.

Потенциальными сервисами, предоставляющими проектные решения являются:

a) opencores.org – библиотека VHDL–моделей IP блоков [25, 119];

b) freemodelfoundry.com – библиотека готовых моделей на VHDL [14];

c) vhdl.org/rassp/vhdl/models – библиотека готовых моделей и алгоритмов на VHDL [29].

4. Интеграция систем через API.

Подобная интеграция возможна только при открытом API сторонней системы. С помощью API можно получить доступ к функциональным возможностям системы, отправляя запросы можно получать некоторые результаты, которые будут использоваться для дальнейшей работы.

4.6. Разработка программно-информационного комплекса: система распределенного проектирования – компьютерная образовательная среда

В настоящее время в образовании, производстве, работе административных структур самого различного уровня наблюдается активное внедрение компьютерных систем обучения, предназначенных для формулирования необходимых компетенций студентов и персонала.

Основными вопросами при разработке и внедрения таких систем является интеграция базы знаний и опыта предприятия (организации), а также его программно-информационного комплекса с системой обучения.

Перспективным направлением организации и разработки компьютерных систем обучения является подход, в основу которого положены интеллектуальные среды.

Основными принципами указанного подхода являются [60].

1. Интеллектуальная среда, представляющая концепцию реализации взаимодействия «человек-техника-среда», организует вокруг человека или группы людей дружественную искусственную микросреду, своего рода «незримого коллективного интеллекта».

2. Персонифицированное обучение, позволяющее формировать уникальную (индивидуальную) траекторию обучения учитывая не только темпоральные значения компетенций, но и ряд других параметров: темп изучения материала и выполнения проектных заданий, перемещения по экрану, предпочтения по виду материала (текст, графика, видео), значимость найденных интернет-источников и др.

3. Изучение опыта и лучших практик предприятия, которые материализованы в виде моделей и документов, являющихся динамическими компонентами темпоральной траектории обучения.

Таким образом, компьютерная система обучения представляется в виде корпоративной образовательной среды (КОС), обобщенная структура которой представлена на рис. 4.10 [2].

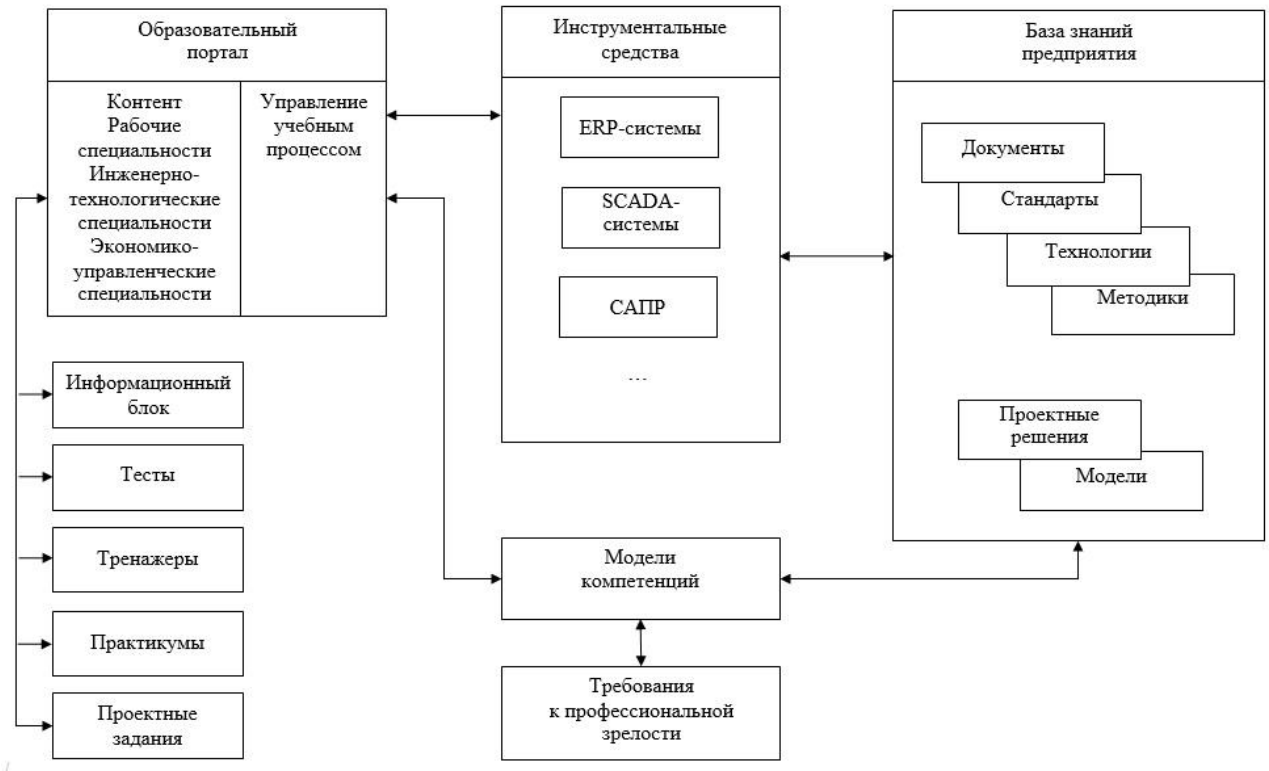


Рис. 4.10. Структура интеллектуальной корпоративной среды обучения

Математическое обеспечение КОС составляют модели и методы, предназначенные для организации персонифицированного практико-ориентированного обучения [61].

1. Модель предметной области, представляющая процессы и объекты, в виде ассоциативного дерева онтологий, которое динамически использует иерархические, упорядоченные и ассоциативные связи, тем самым повышает качество содержания обучения.

2. Модель обучаемого, позволяет создавать адаптивную индивидуальную траекторию и использует нечеткие лингвистические индивидуальные характеристики оценок теоретической и практической степени подготовки обучаемого в предметной области, такие как знания, умения, владение навыками и компетентность.

3. Модель сценария траектории обучения, представляющая процесс обучения и упорядочивающая учебно-методический материал.

4. Интеллектуальный метод диагностики знаний, умений, владения навыками и компетентности обучаемого на базе классификации с помощью нечетких карт Кохонена.

5. Метод управления траекторией обучения и адаптивного планирования, применяющий комплекс моделей «Предметная область», «Обучаемый», «Сценарий» с целью достижения требуемых характеристик.

КОС реализована на базе платформы MOODLE в виде сервисно-ориентированной системы. Сервисно-ориентированная структура образовательного портала показана в табл. 4.1.

Таблица 4.1. Сервисно-ориентированная структура образовательного портала

Сервис	Функция	ПО
Автоматизация деятельности предприятия	Автоматизированное управление обучением	Разработка УлГТУ
Сетевые электронные обучающие системы	Система управления обучением (разработка, хранение, подписка, формирование статистики и т. д.)	LMS Moodle Свободное ПО с открытым кодом
Видеопортал	Хранение и просмотр учебного видеоматериала	CMS Joomla Свободное ПО с открытым кодом
Видеоконференция	Совместная деятельность в режиме реального времени посредством интернет	OpenMeeting Свободное ПО с открытым кодом

Взаимодействие между СРП и сервером КОС

Сетевая организация КОС включает тренажерный сервер, на котором размещается симуляционное программное обеспечение, и рабочие станции, на которых по клиент-серверной технологии проводится обучение проектной деятельности в СРП. В ходе анализа web-технологий, используемых при реализации сетевой КОС и СРП, была сформирована структура взаимодействия двух систем. Данная структура представлена на рис. 4.11.

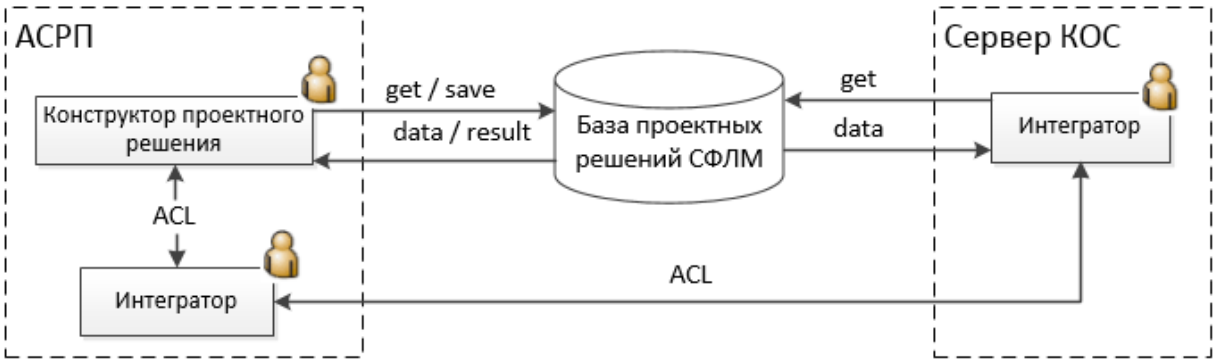


Рис. 4.11. Концептуальная схема взаимодействия АРМ СРП и сервера КОС

На стороне КОС реализовано API (Application Program Interface) представляющее собой интерфейс взаимодействия между клиентом и сервером. API предоставляет возможность работы ресурсов, которые используют потенциал и мощность предоставляющего сайта, а также запуск дополнительных компонентов к ним, расширяющих возможности системы.

Взаимодействие КОС и СРП реализовано по технологии многоагентного взаимодействия. Реализован специализированный агент интеграции, относящийся к типу рефлексных агентов. Функциональные возможности агентов интеграции СРП и КОС представлены в табл. 4.2.

Таблица 4.2. Функциональные возможности агентов интеграции СРП и КОС

Агент интеграции КОС (API)	Агент интеграции СРП
1. Получение обучающего материала по метаданным 2. Получение проектных решений в виде СФЛМ для обучающего материала по его метаданным 3. Получение метаданных обучающего материала	1. Взаимодействие с агентом интеграции КОС 2. Взаимодействие с агентом разработки проектного решения 3. Формирование метаданных из спецификации на проектируемый модуль 4. Запрос на получение обучающих материалов по метаданным

Опишем API системы КОС в виде soap сервиса:

1) Рассмотрим сервис получения метаданным обучающего материала (запрос-ответ)

```
POST /WsKOS/KOS_API.asmx HTTP/1.1
Host: ws.kos.ru
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "https://ws.kos.ru/GetMetaTagsTrainingMaterials"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
	<soap:Body>
		<GetMetaTagsTrainingMaterials xmlns="https://ws.kos.ru/">
			<inParams>
				<metatags>
					<metatag>
						<codeTag>string</codeTag>
						<valueTag>string</valueTag>
					</metatag>
				</metatags>
			</inParams>
		</GetMetaTagsTrainingMaterials>
	</soap:Body>
</soap:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
	<soap:Body>
		<GetMetaTagsTrainingMaterialsResponse xmlns="https://ws.kos.ru/">
			<GetMetaTagsTrainingMaterialsResult>
				<resultCode>int</resultCode>
				<resultString>string</resultString>
				<materials>
					<Material>
						<id_material>int</id_material>
						<type_material>int</type_material>
						<name>string</name>
						<content>string</content>
						<author>string</author>
						<kos_url>string</kos_url>
						<rank>int</rank>
					</Material>
					<Material>
						<id_material>int</id_material>
						<type_material>int</type_material>
```

```
                                        <name>string</name>
                                        <content>string</content>
                                        <author>string</author>
                                        <kos_url>string</kos_url>
                                        <rank>int</rank>
                                    </Material>
                                </materials>
                            </GetMetaTagsTrainingMaterialsResult>
                        </GetMetaTagsTrainingMaterialsResponse>
                </soap:Body>
</soap:Envelope>
```

2) Рассмотрим сервис получения проектных решений для обучающего материала по его метаданным (запрос-ответ)

```
POST /WsKOS/KOS_API.asmx HTTP/1.1
Host: ws.kos.ru
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "https://ws.kos.ru/GetDesignSolutionsByMetaTags"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
        <soap:Body>
                <GetSolutionsByMetaTags xmlns="https://ws.kos.ru/">
                        <inParams>
                                <metatags>
                                        <metatag>
                                                <codeTag>string</codeTag>
                                                <valueTag>string</valueTag>
                                                <rank>int</rank>
                                        </metatag>
                                </metatags >
                        </inParams>
                </GetSolutionsByMetaTags>
        </soap:Body>
</soap:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
        <soap:Body>
                <GetSolutionsByMetaTagsResponse xmlns="https://ws.kos.ru/">
```

```
                    <GetSolutionsByMetaTagsResult>
                        <resultCode>int</resultCode>
                        <resultString>string</resultString>
                        <designSolutions>
                            <DesignSolution>
                                <id_solution>int</id_solution>
                                <view_solution>int</view_solution>
                                <content>string</content>
                                <developer>string</developer>
                                <Task>
                                <name>string</name>
                                <data>string</data>
                                <time>time</time>
                                </Task>
                            </DesignSolution>
                            <DesignSolution>
                                <id_solution>int</id_solution>
                                <view_solution>int</view_solution>
                                <content>string</content>
                                <developer>string</developer>
                                <Task>
                                <name>string</name>
                                <data>string</data>
                                <time>time</time>
                                </Task>
                            </DesignSolution>
                        </designSolutions>
                    </GetSolutionsByMetaTagsResult>
                </GetSolutionsByMetaTagsResponse>
            </soap:Body>
</soap:Envelope>
```

Опишем упрощенный пример метаданных для нескольких функциональных модулей RISC процессора. Учитывая иерархическую структуру проектируемого устройства, которая представлена на рис. 4.12, при формировании спецификации по каждому проектируемому модулю описываются особенности и характеристики данного модуля.

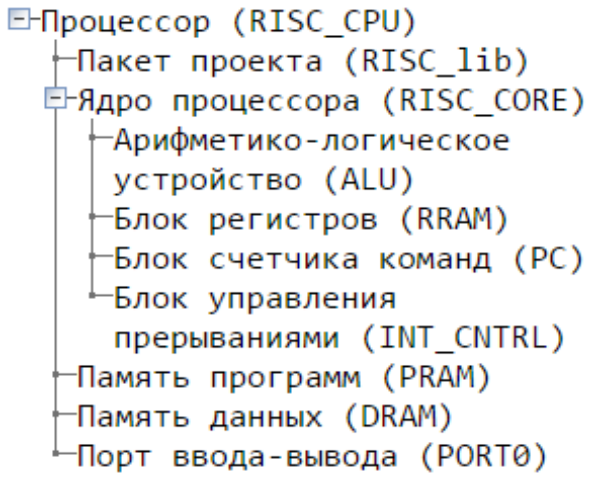


Рис. 4.12. Структура RISC процессора

Для ALU в качестве мета тегов используются его характеристики, такие как тип, разрядность, состав операций, форматы обрабатываемых данных, способ построения и функционирования, способы кодирования данных, быстродействие, надежность, а также его классификация.

1. По способу обработки операндов
    a) последовательные;
    b) параллельные;
    c) последовательно – параллельные.
2. По временным характеристикам
    a) синхронные;
    b) асинхронные.
3. По характеру использования элементов и узлов
    a) блочные;
    b) конвейерные;
    c) многофункциональные.
4. По способу представления данных
    a) с фиксированной запятой;
    b) с плавающей запятой.
5. По использованию систем счисления
    a) двоичная;
    b) двоично-десятичная;
    c) восьмеричная;
    d) шестнадцатеричная;
    e) и т.д.

6. По структуре устройства управления

a) с жесткой логикой устройства управления;

b) с микропрограммным управлением.

Для регистров в качестве мета тегов используются: назначение регистра, разрядность, название, а также их классификация.

1. По типу прима и выдачи информации:

a) С последовательным приемом и выдачей информации – сдвиговые регистры

b) С параллельным приемом и выдачей информации – параллельные регистры

2. По назначению

a) аккумулятор – используется для хранения промежуточных результатов арифметических и логических операций и инструкций ввода-вывода;

b) флаговые – хранят признаки результатов арифметических и логических операций;

c) общего назначения – хранят операнды арифметических и логических выражений, индексы и адреса;

d) индексные – хранят индексы исходных и целевых элементов массива;

e) указательные – хранят указатели на специальные области памяти (указатель текущей операции, указатель базы, указатель стека);

f) сегментные – хранят адреса и селекторы сегментов памяти;

g) управляющие – хранят информацию, управляющую состоянием процессора, а также адреса системных таблиц.

Спецификация и метаданные для остальных функциональных модулей определяются аналогичным образом.

Ранжировка метаданных

Данная операция проводится с целью получения данных отсортированных по релевантности. Ранжирование возлагает определенные требования на

эксперта, который может минимизировать фактор субъективности при выставлении весов для метаданных с целью указания степени важности.

Для упорядочивания или сортировки конечного множества вариантов $A_1,\ldots,A_m$, оцененных по многим количественным критериям $K_1,\ldots,K_n$ без определения числовой ценности вариантов, применяется метод ограниченной пороговой предпочтительности ЭЛЕКТРА (ELimination Et Choix Traduisant la Realité – ELECTRE) [120]. Пороговый подход (Б.Руа, Франция и др.) реализует реляционную модель субъективного выбора.

В методе "ЭЛЕКТРА" разработана процедура многокритериального выбора наиболее предпочтительных объектов, включающая следующие этапы.

1. Для каждого из критериев вводится дискретная шкала возможных значений этого критерия, весовые коэффициенты критериев.

2. Для каждого из критериев строится граф, вершинами которого являются отдельные объекты множества, а дуги указывают на отношение доминирования между объектами в соответствии с данным критерием.

3. С учетом важности критериев и предпочтительности объектов вычисляются матрицы значений специальных коэффициентов, называемых индексами согласия и несогласия.

4. Для каждой пары объектов $(x,y)\in X$ считается установленным отношение превосходства, скажем x над y, если значение соответствующего индекса согласия больше некоторого порогового значения, а индекс несогласия - меньше соответствующего порогового значения.

5. Строится обобщенный граф превосходства, структура которого зависит от выбранных пороговых значений.

4.7. Выводы и рекомендации

1. Разработанные программные средства СРП позволили опробовать на практике предложенные в работе методы и алгоритмы и в результате проведенных экспериментов полностью доказали свою состоятельность.

2. Использование концепции SaaS для организации СРП позволяет проектировщикам всегда работать с актуальной версией системы из любого места, в котором доступен интернет. Предложенная система для проектирования VHDL-объектов способствует созданию и наполнению библиотек VHDL-программ, которые позволяют многократно использовать проектные решения путем модификации данных с учетом требований к новым задачам. Разработка и промышленное внедрение СРП позволит сделать проектирование более эффективным за счет передачи агентам части рутинных операций, например, поиска и синтеза проектных решений, тестирования. Также способствует повышению качества управления процессом коллективного проектирования и сокращению срока разработки.

3. Разработанная архитектура СРП позволило минимизировать требования к аппаратному и программному обеспечению рабочих мест пользователей. Пользователям необходимо следить за актуальной версией браузера для того чтобы корректно работал функционал клиентской части СРП.

4. Применение технологий построения кроссплатформенной системы способствовали расширению круга пользователей посредством обеспечения их полноценной работы независимо от того, с какого устройства зашел и какая операционная система установлена у проектировщика. Адаптивный web интерфейс системы позволяет отображать основные функциональные части СРП на различных устройствах. Система должна быть развернута на сервере с поддержкой php и СУБД MySQL.

5. За счет хранения проектного решения в виде СФЛМ достигается не только возможность организации поиска по критериальным параметрам проектируемого устройства, но и повышается читабельность исходного кода за счет формирования VHDL кода с учетом форматирования текста и пользовательских настроек, таких как стилей отступа, комментариев и т.д. Каждый проектировщик может установить свои настройки по работе в СРП.

6. За счет введения в системе интеграционной составляющей осуществляется возможность поиска проектных решений на сторонних

сервисах, а, следовательно, возлагая эту обязанность на агента, упростить процесс проектирования и уменьшить время получения проектного решения. Использование API системы сторонние разработчики могут дописывать свои пользовательские модули для СРП.

7. Совмещение системы управления задачами и системы формирования проектных решений в виде описаний проектируемых устройств на языке VHDL способствовало уменьшению времени создания законченных VHDL-описаний проектов.

8. Внедрение агентов интеграции в системе СРП и КОС привело к внесению в процесс обучения и разработки новых возможностей, предоставляющих реальные примеры из базы проектных решений по ходу прохождения обучающего курса в КОС и получающих справочную информацию при формировании проектного решения в СРП. Использование API системы для получения проектных решений или обучающих материалов должно осуществляться в соответствии с описанными требования.

9. На программный продукт получено свидетельство РОСПАТЕНТа [65].

ЗАКЛЮЧЕНИЕ

Цель диссертационной работы – сокращения затрат на разработку и повышение формирования качества проектного решения путем реализации методов и средств распределенного проектирования – достигнута.

Основными результатами работы являются.

1. Предложена и реализована архитектура web-ориентированной многоагентной системы распределенного автоматизированного проектирования. Доступ к функционалу системы осуществляется посредствам web-интерфейса с возможностью удаленной работы над проектируемым объектом. Сокращение времени проектирования в среднем составляет 10%.

2. Предложены методы управления задачами в системе распределенного проектирования и формирования библиотек VHDL-программ, позволяющие повысить качество формирования проектных решений за счет эффективной реализации потоков проектных задач, а также наполнения базы проектных решений объектами в виде СФЛМ, поиска наиболее близких проектных решений и их повторного использования.

3. На основе разработанной модели СРП в виде сети Петри получены практические результаты анализа работы СРП и функционирования агентов.

4. Разработано программно-информационное обеспечение СРП.

СПИСОК ЛИТЕРАТУРЫ

1. Afanasev A.N., Khorodov V.S.. Software system of distributed design of complex VHDL objects // Software & Systems. – 2015. – №1 (109). – pp. 42-47.

2. Afanasyev A., Voit N., Voevodin E., Egorova T., Novikova O. INTELLIGENT LEARNING ENVIRONMENTS // Proceedings of INTED2015 Conference 2th-4th March. – 2015, Madrid, Spain. – pp. 4493-4502.

3. Alireza Mokhtar, Omid F.V. Developing a STEP-Compliant Multiagent on an Interoperable and Integrated CAD/CAM Platform // International Journal of Manufacturing Engineering. – 2013. – pp. 1-8.

4. Álvares A.J., Ferreira J.C.E. WebMachining: Implementation of a Collaborative CAD/CAPP/CAM System for E-Manufacturing Through the Internet // The 38th CIRP-International Seminar on Manufacturing Systems. – 2005.

5. Bard A., Finnie E., Forte M., Selvidge W., Stavash J., Tuck M.C., Wedgwood J. Workflow Modeling for Implementing Complex, CAD-Based, Design Methodologies // Proceedings of 2nd Annual RASSP Conference. Arlington. – 1995. – pp. 239-243.

6. Burger, S., Hummel O., Heinisch M. Airbus Cabin Software // IEEE Software. 2013. − №1.− pp. 21−25.

7. Cheremisinov D. The specification of agent interaction in multi-agent systems // Intelligent Information Management. – 2009. – №2. – pp. 65-72.

8. Chih-Hsing Chu, Ping-Han Wu, Yu-Chiung Hsu. Multi-agent collaborative 3D design with geometric model at different levels of detail // Robotics and Computer-Integrated Manufacturing. – 2009. – №25. – pp. 334-347.

9. Chris P. Pro PHP MVC (Expert's Voice in Open Source). – New York: Apress, 2012. – 471 p.

10. Damasevicius R. A subset-based comparison of main design languages // Informacines technologijos ir valuymas. – 2004. – №1 (30). – pp. 49-56

11. David Dornfeld, Paul K. Wright, Shad Roundy, Arvind Rangarajan, Sung-Hoon Ahn. AGENT INTERACTION IN CAD/CAM // Transactions of NAMRI/SME Volume XXIX. – 2001. – pp. 569-575.

12. FIPA. ACL Message Structure Specification [Электронный ресурс] // http://www.fipa.org: The Foundation for Intelligent Physical Agents: http://www.fipa.org/specs/fipa00061/SC00061G.pdf/ (дата обращения: 12.06.2014).

13. FLIR Accelerates Development of Thermal Imaging FPGA. [Электорнный ресурс]// http://www.mathworks.com/tagteam/76668_91997v01_FLIR_UserStory_final.pdf (дата обращения: 27.05.2015).

14. Free Model Foundry. Open Source Simulation Models for System Level Verification**.** [Электронный ресурс] // http://freemodelfoundry.com (дата обращения: 06.06.2015).

15. Friedrich, G., Stumptner M., Wotawa F. Model-Based Diagnosis of Hardware Designs // Artificial Intelligence. − 1999. − pp. 3−39.

16. Git. Documentation. [Электронный ресурс] // http://git-scm.com/doc (дата обращения: 06.06.2015).

17. Green R. Облачные технологии в САПР // CAD/CAM/CAE Observer. – 2010. – №6 (58). – С. 30-33.

18. HDL Coder. [Электорнный ресурс] // http://matlab.ru/products/HDL-Coder (дата обращения: 06.06.2015).

19. Implementation of a MIPS processor in VHDL. [Электорнный ресурс] // http://users.utcluj.ro/~ancapop/labscs/MIPS.pdf (дата обращения: 06.06.2015).

20. Juan Du, Xianguo Yan. Multi-agent System for Process Planning in Step-nc Based Manufacturing // Research Journal of Applied Sciences, Engineering and Technology. – 2012. – №4 (20). – pp. 3865-3871.

21. Liu K., Zhou S. Yang H. Quality metrics of object oriented design for software development and Re-development // Proceedings of the first asia-pacific conference on Quality Software. – 2000. – pp. 127-135.

22. Maciel, R., Albertini B., Rigo S., Araujo G., Azevedo R. A Methodology and Toolset to Enable SystemC and VHDL Co-simulation. // Computer Society Annual Symposium on VLSI (ISVLSI’07). − 2007. – pp. 351−356.

23. McIsaac A., Casaubieilh, F., Benjamin M., Bartley M., Pogodalla F., Rocheteau F., Belhadj M., Eggleton J., Mas G., Barrett G., Berthet C. Functional Verification Methodology of Chameleon Processor // DAC. − USA. − pp. 421−426.

24. Nassehi A., Newman S.T., Allen R.D. The application of multi-agent systems for STEP-NC computer aided process planning of prismatic components // International Journal of Machine Tools & Manufacture. – 2006. – №46. – pp. 559-574.

25. OpenCores. [Электронный ресурс] // http://opencores.org (дата обращения: 06.06.2015).

26. Pablo Santos. Distributed Version Control Systems in the Enterprise [Электронный ресурс] // http://www.infoq.com/articles/DVCS-Enterprise (дата обращения: 06.06.2015).

27. Paul Fanning. From collaboration to integration // RESEARCH & DEVELOPMEN. – 2014. – pp. 19-20.

28. Pierre Bricaud. Reuse Methodology Manual for System-On-A-Chip Designs // Springer; 3rd ed. 2002.

29. RASSP Support Page for VHDL. [Электронный ресурс] // http://vhdl.org/rassp/vhdl/ (дата обращения: 06.06.2015).

30. Read/Write RAM VHDL source code. [Электронный ресурс] // http://www.rfwireless-world.com/source-code/VHDL/read-write-RAM-vhdl-code.html (дата обращения: 02.06.2015).

31. Reis R., Glesner M., Indrusiak L.S. Comparative analysis and application of data repository infrastructure for collaboration-enabled distributed design environments // IEEE COMPUTER SOC. – 2002. – p. 1130.

32. Simple RAM Model. [Электронный ресурс] // https://www.doulos.com/knowhow/vhdl_designers_guide/models/simple_ram_model/ (дата обращения: 02.06.2015).

33. Richard E. Haskell, Darrin M. Hanna. Learning By Example Using VHDL – Advanced Digital Design. LBE Books, Rochester, MI, 2007.

34. Rodgers P., Caldwel N., Clarkson P. J. Managing knowledge in dispersed design companies. Facilitating context-driven design support through multiple perspectives // Artificial Intelligence in Design. 2000. pp. 147-167.

35. Sneha Ambastha. A New Level Of ECAD and MCAD Integration Of Native 3D PCB By Altium. ELECTRONICS ZONE. [Электронный ресурс] // http://electronicsforu.com/electronicsforu/circuitarchives/view_article.asp?sno=1499&title%20=%20A+New+Level+Of+ECAD+and+MCAD+Integration+Of+Native+3D+PCB&id=12387&article_type=8&b_type=new#.VDbLMBZsHm4. (дата обращения: 06.06.2015).

36. S.C.Y. Lu, Cai J., Burkett W., Udwadia F. A methodology for Collaborative Design Process and Conflict Analysis // CIRP Annals – Manufacturing Technology. – 2000. – pp. 69-73.

37. Törlind P., Larsson A. Support for Informal Communication in Distributed Engineering Design Teams // Annals of 2002 Int. CIRP Design Seminar. 2002. Режим доступа: http://pure.ltu.se/portal/files/472844/publication_274.pdf (дата обращения: 06.06.2015).

38. Unified Modeling Language™ (UML®). [Электронный ресурс] // http://www.uml.org/ (дата обращения: 06.06.2015).

39. VHDL code for a 16*16 bit static RAM. [Электронный ресурс] // http://www.edaboard.com/thread92311.html (дата обращения: 02.06.2015).

40. VHDL code for 4-bit ALU. [Электронный ресурс] // http://allaboutfpga.com: Learn FPGA and VHDL Basics. URL: http://allaboutfpga.com/vhdl-code-for-4-bit-alu/ (дата обращения: 02.06.2015)

41. VHDL for FPGA Design/4-Bit ALU. [Электронный ресурс] // http://en.wikibooks.org/wiki/VHDL_for_FPGA_Design/4-Bit_ALU (дата обращения: 02.06.2015)

42. VHDL-Project-16-bit-RISC-Processor. [Электорнный ресурс] // https://github.com/sameersondur/VHDL-Project-16-bit-RISC-Processor/ (дата обращения: 06.06.2015).

43. Workflow Patterns. [Электронный ресурс] // www.workflowpatterns.com (дата обращения: 20.08.2014).

44. 4-bit ALU using VHDL. [Электронный ресурс] // http://www.eeweb.com/project/mohd_kashif/4-bit-alu-using-vhdl (дата обращения: 02.06.2015).

45. 16 bit microprocessor design using vhdl. [Электорнный ресурс] http://dspace.jdvu.ac.in/bitstream/123456789/29299/1/Acc.%20No.%20DC%201724.pdf (дата обращения: 06.06.2015).

46. Адамов А.З. Разработка и исследование распределенных Web-ориентированных архитектур САПР: Дис. канд. техн. наук (рук. Глушань В.М.) – Таганрог, 2003. – 195 с.

47. Анисимов В.И. Методы построения систем автоматизированного проектирования на основе Интернет-технологий и компактной обработки разреженных матриц / Гридин В.Н., Анисимов В.И. // Информационные технологии в проектировании и производстве. – 2009. – №1.

48. Анисимов В.И. Архитектура схемотехнических САПР со встроенным броузером / Гридин В.Н., Анисимов В.И., Ларистов Д.А. // Автоматизация в промышленности. – 2009. – №11.

49. Анисимов В.И. Технология построения системы информационной поддержки распределенных процессов проектирования / Гридин В.Н., Анисимов В.И., Башкатов А.С. // Автоматизация в промышленности – 2010. – №2.

50. Анисимов В.И. Методика построения встраиваемых подсистем организации процесса распределенного проектирования / Анисимов В.И., Башкатов А.С. // Известия СПбГЭТУ. – 2011. – № 1.

51. Анисимов Д.А. Платформенно зависимые и независимые системы: сравнительная оценка и методы взаимодействия [Текст] / Анисимов Д.А. // Материалы XIV международной конференции «Современное образование: содержание, технологии, качество». Том1. – СПб., СПбГЭТУ, 2008. – С. 212-213.

52. Анисимов Д.А. Описание электронных аналоговых схем с помощью иерархической xml структуры [Текст] / Анисимов Д.А. // Сборник докладов студентов, аспирантов и молодых ученых – СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2011. – С. 126-132.

53. Анисимов Д.А., Дмитревич Г.Д., Гридин В.Н. Распределенные системы автоматизированного проектирования на основе сервис-ориентированной архитектуры // Труды международной научно-технической конференции 2011 «Информационные технологии и математическое моделирование систем» – М.: ЦИТП РАН, 2011. – С. 48-50.

54. Анисимов Д.А. Гридин В.Н., Дмитревич Г.Д. Построение систем автоматизированного проектирования на основе Web-сервисов // Автоматизация в промышленности. – 2011. – №1. – С. 9-12.

55. Анисимов Д.А., Гридин В.Н., Дмитревич Г.Д. Построение систем автоматизированного проектирования на основе Web-технологий // Информационные технологии. – 2011. – №5. – С. 23-27.

56. Анисимов Д.А., Гридин В.Н., Дмитревич Г.Д. Построение веб-сервисов систем автоматизации схемотехнического проектирования // Информационные технологии и вычислительные системы. – 2012. – №4. – С. 79-84.

57. Арлазаров В.Л., Емельянов Н.Е. От баз данных к базам знаний (объекты, формы, содержание) // Труды ИСА РАН. – 2006. – Т. 23. – С. 6-17.

58. Асратян Р.Э., Лебедев В.Н., Дмитриев Р.И. Интернет и распределенные многоагентные системы // Перспективные информационные технологии и концепции: Ленанд, 2007. – 72 с.

59. Афанасьев А.Н. Методология графо-аналитического подхода к анализу и контролю потоков работ в автоматизированном проектировании сложных компьютеризованных систем // Вестник Ульяновского государственного технического университета. – 2011. – №3 (55). – С. 48-52.

60. Афанасьев А.Н. Разработка методов и средств многоагентного распределенного автоматизированного проектирования структурно-функциональных лингвистических моделей вычислительных устройств //

Информатика, моделирование, автоматизация проектирования: сборник научных трудов Всероссийской школы-семинара. – Ульяновск: УлГТУ, 2014. – С. 191-197.

61. Афанасьев А.Н., Войт Н.Н., Гульшин В.А. Концепция организации и реализация корпоративной образовательной среды промышленного предприятия // Труды Конгресса по интеллектуальным системам и информационным технологиям «IS&IT'13». Научное издание в 4-х томах. – М.: Физматлит, 2013. – Т.2. – С. 252-257.

62. Афанасьев А.Н., Игонин А.Г. Обработка лингвистических структурно-функциональных моделей на основе нейросемантических сетей: монография. Ульяновск: УлГТУ, 2007. – 227 с.

63. Афанасьев А.Н., Игонин А.Г. Применение нейросемантического подхода для анализа и синтеза функциональных моделей в системах проектирования // Вестник Ижевского государственного технического университета. – 2007. – №1. – С. 66-69.

64. Афанасьев А.Н., Хородов В.С. Модель управления потоками работ в системе распределенного проектирования VHDL-объектов // Технические науки – от теории к практике: сборник статей по материалам XXXVII международной научно-практической конференции. №8(33). Новосибирск: СибАК. – 2014. – С. 6-11.

65. Афанасьев А.Н., Хородов В.С. Web-ориентированный транслятор VHDL описаний в шаблоны структурно-функциональных лингвистических моделей. РОСПАТЕНТ: свидетельство № 2015610356 от 12 января 2015 г.

66. Афанасьев А.Н., Хородов В.С. Агентный подход к построению системы распределенного проектирования VHDL-объектов и её моделирование на базе цветных сетей Петри // Известия Волгоградского государственного технического университета. – 2014. Т.22. – №25(152). – С. 47-53.

67. Афанасьев А.Н., Хородов В.С. Моделирование распределенной системы проектирования VHDL-объектов // Информатика и вычислительная

техника: сборник научных трудов Всероссийской научно-технической конференции. – Ульяновск: УлГТУ, 2014. – С. 21-33.

68. Афанасьев А.Н., Хородов В.С. Разработка и моделирование распределенной системы проектирования VHDL-объектов // Автоматизация. Современные технологии. – 2015. – №4. – С. 34-40.

69. Афанасьев А.Н., Хородов В.С. Распределённое проектирование структурно-функциональных моделей представленных на языке VHDL // Вестник Ульяновского государственного технического университета. – 2014. – №2 (66). – С. 41-45.

70. Афанасьев А.Н., Хородов В.С. Распределенное проектирование структурно-функциональных лингвистических моделей // Труды Конгресса по интеллектуальным системам и информационным технологиям «IS&IT’14». – М.: Физматлит, 2014. – Т. 1. – С. 331-337.

71. Афанасьев А.Н., Хородов В.С. Разработка модели управления задачами в системе распределенного проектирования VHDL-программ // Вестник Ульяновского государственного технического университета. – 2015. – №2. – С. 33-38.

72. Афанасьев А.Н., Хородов В.С. Система автоматизированного проектирования структурно-функциональных лингвистических моделей // Автоматизация процессов управления. – 2014. – №4(38). – С. 92-98.

73. Афанасьев А.Н., Хородов В.С. Технологии распределенного проектирования VHDL-объектов // Вестник Ижевского государственного технического университета. – 2014. – №4 (64). – С. 131-134.

74. Афанасьев М.Я., Яблочников Е.И., Саломатина А.А., Алешина Е.Е. Применение многоагентных технологий для реализации системы управления виртуальным предприятием // Научно-технический вестник Санкт-Петербургского государственного университета информационных технологий, механики и оптики. 2011. № 5 (75). С. 105-110.

75. Берчун Ю.В. Язык описания электронной аппаратуры VHDL: Методические указания. М.: МГТУ им. Н.Э. Баумана, 2006. – С. 39-41.

76. Бирюков А. Средства автоматизации проектной деятельности // Бизнес & информационные технологии. – 2014. – №9 (42). Режим доступа: http://bit.samag.ru/archive/article/1411 (дата обращения: 06.06.2015).

77. Вичугова А.А., Вичугов В.Н., Цапко Г.П. Формальная модель структуры взаимосвязей разнотипных объектов проектирования // Известия Томского политехнического университета. – 2013. – Т. 322. – № 5. – С. 164-169.

78. Владимиров А.В. Моделирование взаимодействия агентов в многоагентной системе с помощью цветных сетей Петри и нечеткой логики // Программные продукты и системы. – 2014. – №1. – С. 44-50.

79. Габалин А.В., Разбегин В.П. Анализ и синтез структуры Workflow-систем // Материалы конференции 11-я международная конференция «Системы проектирования технологической подготовки производства и управления этапами жизненного цикла промышленного продукта» (CAD/CAM/PDM-2011). М.: ИПУ РАН, 2011. – С. 93-96.

80. Гик Ю. Анализ методологий сервисно-ориентированной архитектуры (СОА) // Конференция «Разработка ПО 2012» CEE-SECR 2012. Режим доступа: http://2012.secr.ru/lang/ru-ru/talks/service-oriented-architecture-methodologies-analysis (дата обращения: 06.06.2015).

81. Глушань В.М., Адамов А.З. Технология проектирования сетевых САПР // Труды международных конференций IEEE AIS’02 и ICAD-2002, М.: Физматлит, 2002. – С. 456-461.

82. Глушань В.М., Иванько Р.В., Лаврик П.В., Рыбальченко М.В. Сравнительный анализ эффективности распределенных САПР // Труды Международных научно-технических конференций “Интеллектуальные системы” (AIS’06) и “Интеллектуальные САПР” (CAD-2006). – М.: Физматлит, 2006. – Т.2. – С. 33-39.

83. Глушань В.М., Лаврик П.В. Исследование клиент-серверной модели распределенной САПР электорнных схем // Известия ЮФУ. Технические науки.

Тематический выпуск "Интеллектуальные САПР" – Таганрог: ТТИ ЮФУ, 2009. – №4. – С. 77-81.

84. Глушань В.М., Лаврик П.В. Исследование эффективности распределенных САПР СБИС // VIII Всероссийская научная конференция молодых ученых, аспирантов и студентов “Информационные технологии, системный анализ и управление”. – Таганрог: ТТИ ЮФУ, 2010. – С. 381-383.

85. Гольфанд И.Я., Хлебутин П.С. Оценка трудозатрат разработки программной компоненты // Труды ИСА РАН. – 2005. – Т.15. – С. 125-135.

86. Городецкий В.И. Многоагентные системы: современное состояние исследований и перспективы применения // Новости искуственного интеллекта. – 1996. – № 1. – С. 44-49.

87. Григорьев А.В., Кошелева Д.А. ИНТЕЛЛЕКТУАЛИЗАЦИЯ ПРОЦЕССА ПРОЕКТИРОВАНИЯ АППАРАТУРЫ СРЕДСТВАМИ ЯЗЫКА VHDL // международная научно-техническая конференция "МОДЕЛИРОВАНИЕ И КОМПЬЮТЕРНАЯ ГРАФИКА". 2005. С. 182-188. Режим доступа: http://ea.donntu.org:8080/jspui/bitstream/123456789/15794/1/182-188.pdf (дата обращения: 06.06.2015).

88. Григорьев А.В., Грищенко Д.А. Классификация источников знаний о методиках проектирования VHDL-программ в инструментальной оболочке по созданию интеллектуальных САПР в режиме «глупого» эксперта // Четверта міжнародна науково-технічна конференція “Моделювання та комп’ютерна графіка”. Донецьк, ДонНТУ, 2011. – С. 67-73.

89. Григорьев А.В., Грищенко Д.А. Анализ методов извлечения знаний о методиках проектирования из VHDL-текстов // Наукові праці ДонНТУ. Серія “Інформатика, кібернетика та обчислювальна техніка”. – 2012. – №16(204). – С.143-149.

90. Григорьев А.В., Грищенко Д.А. Анализ средств автоматизации построения VHDL-программ // Наукові праці ДонНТУ. Серія “Інформатика, кібернетика та обчислювальна техніка”. – 2011. – 14(188). – С. 262-269.

91. Грушвицкий Р.И., Мурсаев А.Х., Угрюмов Е.П. Проектирование систем на микросхемах с программируемой структурой / 2-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2006. – 608 с.

92. Денисов А. Несколько советов по проектированию цифровых устройств на VHDL для ПЛИС // Компоненты и технологии. – 2009. – № 12. – С. 48-51.

93. Дружинин Е.А., Елисеев Д.Н. Развитие систем автоматизированного проектирования // Двигатель. №3(45). 2006. Режим доступа: http://engine.aviaport.ru/issues/45/page56.html (дата обращения: 06.06.2015).

94. Евтушенко Н., Немудров В., Сырцов И. Методология проектирования систем на кристалле. Основные принципы, методы, программные средства // Электроника: наука, технология, бизнес. – 2003. – № 6. – С. 7-11.

95. Ехлаков Ю.П., Тарасенко В.Ф., Жуковский О.И., Сенченко П.В., Гриценко Ю.Б. Цветные сети Петри в моделировании социально-экономических систем // Доклады ТУСУРа. – 2013. – №3(29). – С. 83-92.

96. Зотов В.Ю. Проектирование встраиваемых микропроцессорных систем на основе ПЛИС фирмы XILINX. – М.: Горячая линия – Телеком, 2006. – 520 с.

97. Зубакин И.А., Фахми Ш.С., Шагаров С.С. Аппаратно-программное проектирование сложных функциональных блоков с использованием систем на кристалле // Научно-технический вестник Санкт-Петербургского государственного университета информационных технологий, механики и оптики. – 2010. – № 2(66). – С. 90-98.

98. Ивченко В.Г. Применение языка VHDL при проектировании специальных СБИС. Т.: ТГРУ, 2000. – С. 1-47.

99. Камаев В.А. Концептуальное проектирование. Развитие и совершенствование методов: монография. [коллективная] / Камаев В.А., Бутенко Л.Н., Дворянкин А.М., Фоменков С.А., Бутенко Д.В., Давыдов Д.А., Заболеева-Зотова А.В., Жукова И.Г., Кизим А.В., Колесников С.Г., Костерин В.В., Петрухин А.В., Набока М.В.. – М.: Машиностроение-1, 2005. – 360 с.

100. Карякин А.Т., Хакулов А.М. Основные возможности САПР Altium Designer // Молодой ученый. – 2014. – № 2. – С. 146-148.

101. Кайнер М. Логика интеграции ECM-системы и сторонних облачных сервисов. [Электронный ресурс] // http://www.docflow.ru/news/analytics/detail.php?ID=29327 (дата обращения: 06.06.2015).

102. Коваль А.А. Применение цветных сетей Петри для моделирования сценария работы приложений // Научная сессия МИФИ-2005. Т.2 Технологии разработки программных систем. Информационные технологии. – 2005. – С. 99-100.

103. Ковалев С.М., Ковалев В.М. Современные методологии и стандарты описания бизнес-процессов: преимущества, недостатки и области применения // Справочник экономиста. – 2006. – №11. Режим доступа: http://www.betec.ru/index.php?id=06&sid=104 (дата обращения: 06.06.2015).

104. Коновалов А.В. Агентный подход в сапр ковки коротких поковок // Программные продукты и системы. – 2011. – №1. – С. 148-152.

105. Котенко И.В. Многоагентная модель поддержки принятия решений при кооперативной работе проектировщиков // Перспективные информационные технологии и интеллектуальные системы. – 2001. – №1 (5). – С. 10-21.

106. Кулямин В.В. Компонентные технологии и разработка распределенного ПО. Иитуит. Национальный открытый университет. [Электронный ресурс] // http://www.intuit.ru/studies/courses/64/64/lecture/1888?page=4 (дата обращения: 06.06.2015).

107. Курдюков А. А. Интеллектуальные агенты и их применение в инженерном проектировании // Матер. конф. и выставки «Системы проектирования, технологической подготовки производства и управления этапами жизненного цикла промышленного продукта. CAD/CAM/PDM-2001». М., 2001.

108. Лазарев И.В., Сухорослов О.В. Использование workflow-методологии для описания процесса распределенных вычислений. Труды ИСА РАН. – 2005. – Т. 14. – С. 26-70.

109. Лисяк В.В., Лисяк Н.К. Программные продукты САПР устройств с программируемой логикой. // Известия ЮФУ. Технические науки. – Таганрог: ТТИ ЮФУ, 2014. – №7 (156). – С. 93-101.

110. Малиновский М.Л., Фурман И.А., Аллашев А.Ю., Конищева А.П., Святобатько А.В. Концепция создания табличных языков описания аппаратуры // Радіоелектронні і комп'ютерні системи. – 2010. – № 6. – С. 289–291.

111. Малиновский М.Л., Конищева А.П., Аленин Д.А., Пушкар А.Н. Сравнительный анализ табличных и текстовых средств проектирования аппаратуры цифровых систем // Вісник Харківського національного технічного університету сільського господарства імені Петра Василенка. – 2013. – С.39-40.

112. Малышева Е.Ю. Учебно-методический комплекс по дисциплине "Распределённые информационные системы". – Тольятти: ПВГУС, 2013. Режим доступа: http://www.vspu.ac.ru/~chul/gosvpo/files/pi/raspr_inf_syst_1.pdf (дата обращения: 06.06.2015).

113. Морозов С., Соколов С. Аспекты эволюции субмикронной микроэлектроники – взгляд изнутри // Chip News. 2004. № 5. С. 4–13.

114. Мохсен А.А.А., Дмитревич Г.Д., Ларистов А.И. Архитектура WEB-ориентированных САПР // Информационно-управляющие системы. – 2010. – №5 (48). – С. 20-23.

115. Мутовкина. Н.Ю., Кузнецов. В.Н., Клюшин. А.Ю., Палюх. Б.В. Нечёткие методы согласованного управления в многоагентных системах // Вестник ТГТУ. – 2013. – Т.19. – №4. – С. 740-750.

116. Муйземнек О.Ю., Коновалов А.В., Гагарин П.Ю. Мультиагентный графический редактор сапр ковки // Программные продукты и системы. – 2011. – №2. – С. 148-151.

117. Немудров В., Мартин Г. Системы-на-кристалле. Проектирование и развитие. – М.: Техносфера, 2004. – 216 с.

118. Непомнящий О.В., Хныкин А.В. Анализ проектирования вычислительных систем на кристалле // Исследования Науко града. – 2012. – С. 42-46.

119. Обзор OSHW-проекта и сообщества OpenCores. [Электронный ресурс]. http://nerohelp.info/1884-opencores.html (дата обращения: 06.06.2015).

120. Петровский А.Б. Теория и методы принятия решений. – М.: Издательский центр «Академия», 2009. – 400 с.

121. Погребинский А., Павлов А. Сравнительный анализ CAD/CAM-систем // САПР и графика. 2000. № 8. Режим доступа: http://www.sapr.ru/article.aspx?id=7694&iid=313 (дата обращения: 06.06.2015).

122. Полозов Р. Современный программный комплекс. Технология CORBA // Системы безопасности. №5. 2005. Режим доступа: http://house-control.org.ua/article/3856/p--polozov--sovremennyy-programmnyy-kompleks--tehnologiya-corba-/ (дата обращения: 06.06.2015).

123. Пранович В.И. От PCAD к Altium Designer // EDAExpress. – 2007. – №15.

124. Рассел С., Норвиг П. Искусственный интеллект: современный подход. – 2-е изд. – М.: Вильямс, 2006. – 1408 с.

125. Романников Д.О., Марков А.В. Об использовании программного пакета CPN Tools для анализа сетей Петри // Сборник научных трудов НГТУ. – 2012. – №2(68). – С. 105-116.

126. Рушди А. Амара. Методы распределенного поиска информации в интернет. Режим доступа: http://ict.edu.ru/ft/001762/2001_3-4_111-119.pdf (дата обращения: 06.06.2015).

127. Сабунин А.Е. Altium Designer. Новые решения в проектировании электронных устройств. –М.: Салон-Пресс, 2009. – 432 с.

128. Сапегин С.В. Проектирование архитектуры информационных систем на основе SOA // Вестник ВГУ, Серия: Системный анализ и информационные технологии. – 2010. – №1. – С. 80-84.

129. Сергиенко А.М. VHDL для проектирования вычислительных устройств. – К.: –"ДиаСофт". – 2003. – 210 с.

130. Слама Д., Гарбис Д., Рассел П. Корпоративные системы на основе CORBA. – М.: Издательский дом «Вильямс», 2000. – 368 с.

131. Смирнов А.В., Шереметов Л.Б. Многоагентная технология проектирования сложных систем // Автоматизация проектирования. №03. 1998. Режим доступа: http://www.osp.ru/ap/1998/03/13031692/ (дата обращения: 06.06.2015).

132. Смирнов А.В., Шереметов Л.Б. Организация взаимодействия агентов в многокомпонентных САПР // Автоматизация проектирования, №02. 1999. Режим доступа: http://www.osp.ru/ap/1999/02/13031729/ (дата обращения: 06.06.2015).

133. Смирнов А.В., Шереметов Л.Б. Многоагентная технология проектирования сложных систем. Окончание // Автоматизация проектирования. №01. 1999. Режим доступа: http://www.osp.ru/ap/1999/01/13031715/ (дата обращения: 06.06.2015).

134. Смирнов А.В., Юсупов Р.М. Технология параллельного проектирования: основные принципы и проблемы внедрения // Автоматизация проектирования. 1997. №2. С. 50-55. Режим доступа: http://www.osp.ru/ap/1997/02/13031618/ (дата обращения: 06.06.2015).

135. Смолов С.А., Камин А.С. Метод построения расширенных конечных автоматов по HDL-описанию на основе статического анализа кода // Научно-технические ведомости СПбГПУ. Информатика. Телекоммуникации. Управление. – 2015. – №1 (212). – С. 60-73.

136. Суворова Е.А., Шейнин Ю.Е. Проектирование цифровых систем на VHDL. – СПб.: БХВ-Петербург, 2003. – 576 с.

137. Станкевич В. CORBA вчера, сегодня, завтра // Software. 2008. №2. Режим доступа: http://old.kv.by/index2008021108.htm (дата обращения: 06.06.2015).

138. Стариков А.В., Бакулина Н.Н., Бунаков П.Ю., Каскевич Н.В. Создание единой управляемой среды проектирования в комплексной САПР корпусной мебели // САПР и графика. – 2010. – №5. Режим доступа: http://www.sapr.ru/article.aspx?id=21402&iid=975 (дата обращения: 06.06.2015).

139. Старых В.А. Открытая территориально-распределенная система управления информационными ресурсами. Известия Орловского государственного технического университета. Серия: Информационные системы и технологии. – 2010. – Т. 6. – № 62. – С. 90-99. Режим доступа: http://www.hse.ru/pubs/lib/data/access/ram/ticket/3/14335892669ecec9a6e8c4538c1a1a67c812eb28f0/article1p1-%D0%B2_%D0%B6%D1%83%D1%80%D0%BD%D0%B0%D0%BB_%D0%9E%D1%80%D1%91%D0%BB.pdf (дата обращения: 06.06.2015).

140. Степанов Е.О., Ярцев Б.М. Учебно-методическое пособие по дисциплине «Архитектуры и технологии разработки распределенного программного обеспечения». Режим доступа: http://books.ifmo.ru/file/pdf/417.pdf (дата обращения: 06.06.2015).

141. Таненбаум Э., М. ван Стеен. Распределенные системы. Принципы и парадигмы. Питер. – 2003. – 197 с.

142. Тарасов В.Б. От многоагентных систем к интеллектуальным организациям: философия, психология, информатика. – М.: Эдиториал УРСС, 2002. – 352 с.

143. Тарасов В.Б. Агенты, многоагентные системы, виртуальные сообщества: стратегическое направление в информатике и искусственном интеллекте // Новости искусственного интеллекта. – 1998. – №2. – С. 5-63.

144. Тарханов А.И. Интеграция Системы Электронных Выплатных Дел со сторонними приложениями // Труды ИСА РАН. – Т.23. – 2006.

145. Трахтенгерц Э.А. Повышение надежности последовательно параллельного проектирования сложных технических объектов // Автомат. и телемех. – 1994. – №5. – С. 138–157.

146. Тучков А. Создание корпоративной САПР: "как совместить желание и возможности" // САПР и графика. – 2000. – №10. – С. 2-5. Режим доступа: http://esg.spb.ru/files/articles/75/Wr92g2JY1L.pdf (дата обращения: 06.06.2015).

147. Федотов А.М. // Тр. Междунар. конф. по вычислительной математике МКВМ-2004. Новосибирск: Изд-во ИВМиМГ СО РАН, 2004. – С. 132−143. Режим доступа: http://www-sbras.nsc.ru/ws/dicr/8046/rep8046.pdf (дата обращения: 06.06.2015).

148. Хородов. В.С. WEB-ориентированный подход к построению САПР // Информатика, моделирование, автоматизация проектирования: сборник научных трудов Всероссийской школы-семинара. – Ульяновск: УлГТУ, 2013. – С. 184-190.

149. Хородов В.С. Проектирование многоагентной распределённой системы создания проектных решений с использованием структурно-функциональных лингвистических моделей // Объектные системы: материалы VIII международной научно-практической конференции. – Ростов-на-Дону, 2014. – С. 19-23.

150. Хородов В.С. Разработка методов синтеза распределенного проектирования структурно-функциональных лингвистических моделей // Системы проектирования, моделирования, подготовки производства и управление проектами CAD/CAM/CAE/PDM: сборник статей по материалам VIII Международной научно-практической конференции. Пенза: Приволжский Дом Знаний, 2014. – С.83-86.

151. Хородов В.С. Разработка методов и средств многоагентного распределенного автоматизированного проектирования структурно-функциональных лингвистических моделей вычислительных устройств // Информатика, моделирование, автоматизация проектирования: сборник научных трудов Всероссийской школы-семинара. – Ульяновск: УлГТУ, 2014. – С. 191-197.

152. Хородов В.С., Игонин А.Г. Технологии распределённого проектирования // Вестник Ульяновского государственного технического университета. – 2014. – №1. – С. 55-59.

153. Целищев Е.С., Пантелеев Е.Р., Ильичев Н.Б., Пекунов В.В., Первовский М.А. Реализация распределенного проектирования в САПР AutomatiCS на базе технологии XML // CADmaster. – 2002. – №4(14). Режим доступа: http://www.cadmaster.ru/magazin/articles/cm_14_automatics_xml.html (дата обращения: 06.06.2015).

154. Черников Б.В., Поклонов Б.Е. Оценка качества программного обеспечения. М.: «ФОРУМ» – ИНФРА-М. 2012. Режим доступа: http://www.hse.ru/data/2012/05/06/1250541039/%D0%9A%20%D0%9E%D0%9A%D0%9F%D0%9E.pdf (дата обращения: 06.06.2015).

155. Чейт С. Преобразование Web-приложения в мультитенантное решение SaaS [Электронный ресурс] // www.ibm.com/developerworks/: ресурс IBM для разработчиков и ИТ-специалистов. URL: http://www.ibm.com/developerworks/ru/library/cl-multitenantsaas/ (дата обращения: 13.06.2014).

156. Шагурин И, Родионов А. IP-блок для реализации функций управления в составе СБИС класса «система на кристалле» Время электорники. [Электронный ресурс] // http://www.russianelectronics.ru/leader-r/review/optic/350/doc/551/ (дата обращения: 13.06.2014).

157. Шостак И.В. Управление сложными объектами в реальном времени на основе динамических экспертных систем // Авиационно-космическая техника и технология. Харьков, 1999. – №10. – С. 204-210.

ПРИЛОЖЕНИЯ

Приложение 1. МОДЕЛЬ ПОДСИСТЕМЫ ПОИСКА (SSE) НА БАЗЕ СЕТИ ПЕТРИ

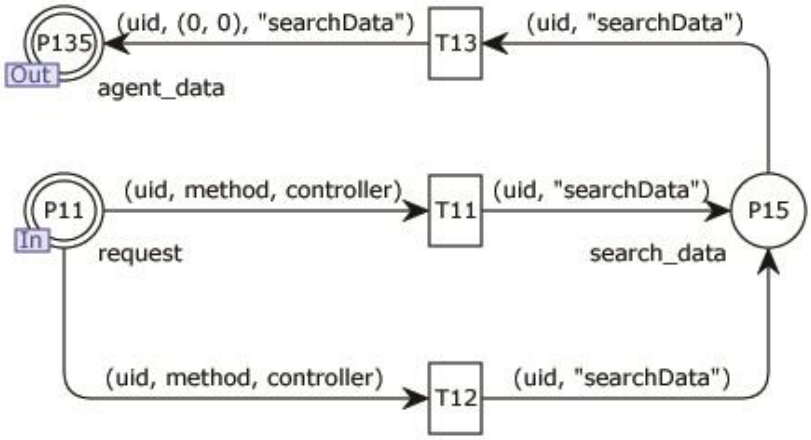


Рис. П1.1. Модель подсистемы поиска (SSE) на базе сети Петри

Таблица П1.1. Описание позиций и переходов модели подсистемы SSE

Обозначение элемента	Описание
P11	Осуществлен переход на страницу с выбором вида поиска
P15	Ввод данных завершен
P135	Запрос на поиск передан
T11	Ввод текста для поиска по ключевым словам. (Упрощенный поиск)
T12	Ввод текста и критериальных параметров. (Расширенный поиск)
T13	Начать поиск

Приложение 2. МОДЕЛЬ ПОДСИСТЕМЫ УПРАВЛЕНИЯ ПРОЕКТАМИ (SMP) НА БАЗЕ СЕТИ ПЕТРИ

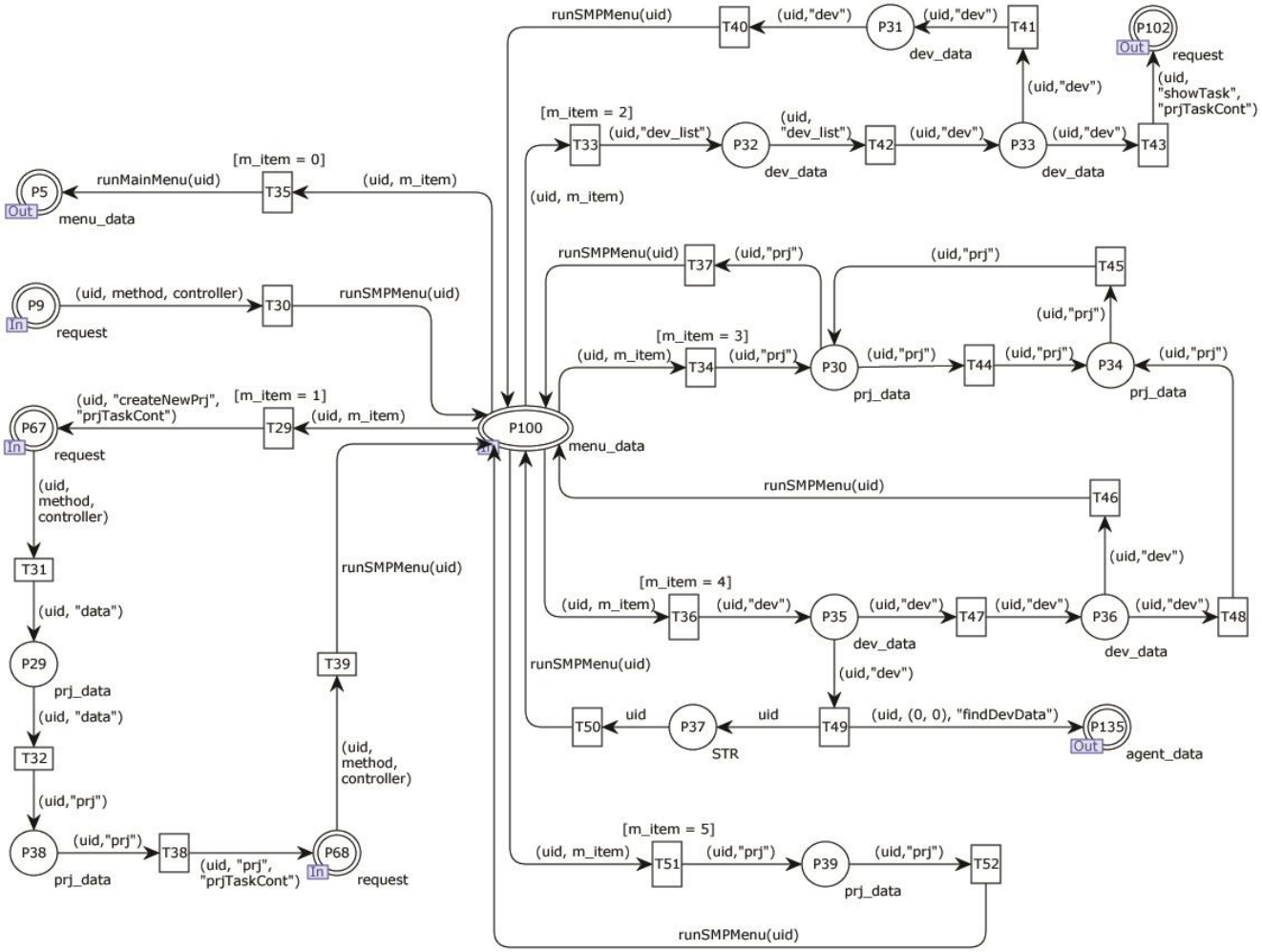


Рис. П2.1. Модель подсистемы управления проектами (SMP) на базе сети Петри

Таблица П2.1. Описание позиций и переходов модели подсистемы SMP

Обозначение элемента	Описание
P9	Осуществлен переход на страницу управления проектами
P67	Осуществлен переход на страницу создания проекта
P29	Данные введены
P38	Проект успешно сохранен
P68	Осуществлен вход в проект
P100	Выбран элемент меню
P5	Переход выполнен
P30	Осуществлен переход на страницу данными проекта
P34	Данные отредактированы
P35	Осуществлен переход на страницу назначения/смены проектировщика
P36	Результат выбора

P37	Продолжить работу в системе
P135	Запрос отправлен
P39	Осуществлен переход на страницу с информацией по проекту
P32	Процесс формирования списка проектировщиков
P33	Проектировщик выбран
P31	Открыта форма с данными о проектировщике
P102	Переход выполнен
T30	Выбор элемента меню
T29	Переход к созданию проекта
T35	Переход к главному меню
T31	Ввод данных для создания проекта
T32	Сохраняем проект
T38	Входим в проект
T39	Выбор элемента меню
T33	Просмотр команды проектировщиков
T37	Выбор элемента меню
T34	Редактирование данных проекта
T36	Назначение/смена проектировщика
T50	Выбор элемента меню
T51	Просмотр сводной информации по проекту
T52	Выбор элемента меню
T49	Запрос на поиск подходящего проектировщика для проекта
T47	Выбор проектировщика из списка доступных в системе
T48	Проектировщик найден. Добавление его к команде проекта
T46	Выбор элемента меню
T44	Редактирование данных
T45	Открытие формы с измененными данными
T42	Выбор проектировщика
T43	Переход к просмотру списка проектных задач с фильтрацией по проекту и проектировщику
T41	Просмотр информации
T40	Выбор элемента меню

Приложение 3. МОДЕЛЬ ПОДСИСТЕМЫ ЛИЧНОГО КАБИНЕТА (SPC) НА БАЗЕ СЕТИ ПЕТРИ

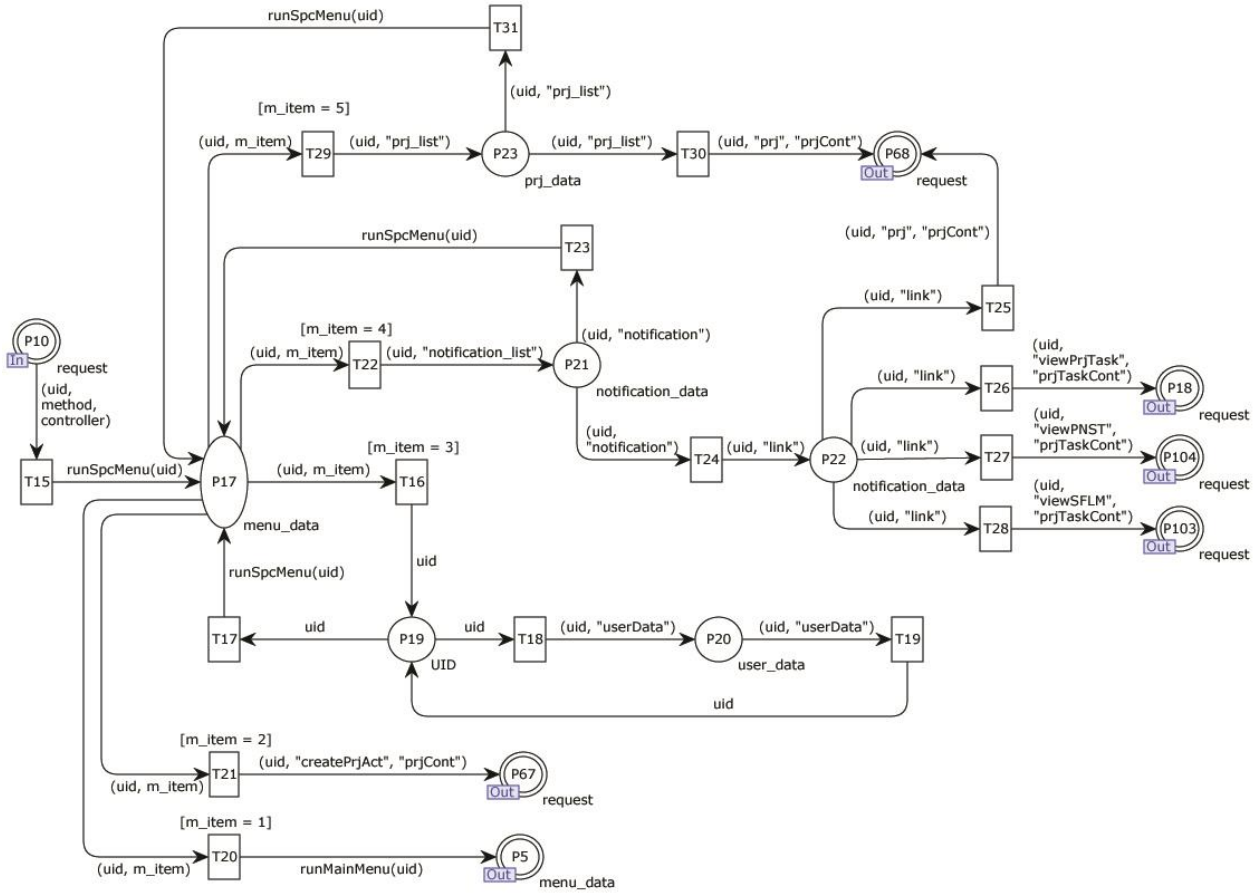


Рис. П3.1. Модель подсистемы личного кабинета (SPC) на базе сети Петри

Таблица П3.1. Описание позиций и переходов модели подсистемы SPC

Обозначение элемента	Описание
P10	Осуществлен переход в личный кабинет
P17	Выбран элемент меню
P19	Осуществлен переход на страницу с личными данными
P21	Сформирован список уведомлений
P23	Сформирован список проектов
P22	Ссылка выбрана
P20	Данные отредактированы
P67	Переход выполнен
P5	Переход выполнен
P68	Проект выбран
P18	Осуществлен переход к странице управления результатами поиска СФМ
P104	Осуществлен переход к проектной задаче
P103	Осуществлен переход к странице управления результатами поиска ПССЗ

T15	Выбор элемента меню
T17	Выбор элемента меню
T21	Переход к форме создания проекта
T20	Переход к главному меню
T16	Просмотр /Редактирование личных данных
T18	Редактирование данных
T19	Открытие формы с измененными данными
T22	Просмотр уведомлений
T29	Просмотр своих проектов
T31	Выбор элемента меню
T23	Информирование об отсутствии уведомлений. Выбор элемента меню
T30	Выбор проекта
T24	Просмотр уведомления. С возможностью перехода по ссылкам
T25	Переход к проекту
T26	Переход к задаче
T27	Переход к результатам поиска ПССЗ
T28	Переход к результатам поиска СФЛМ

Приложение 4. МОДЕЛЬ ПОДСИСТЕМЫ КОНСТРУКТОРА ПРОЕКТНОГО РЕШЕНИЯ (SCPS) НА БАЗЕ СЕТИ ПЕТРИ

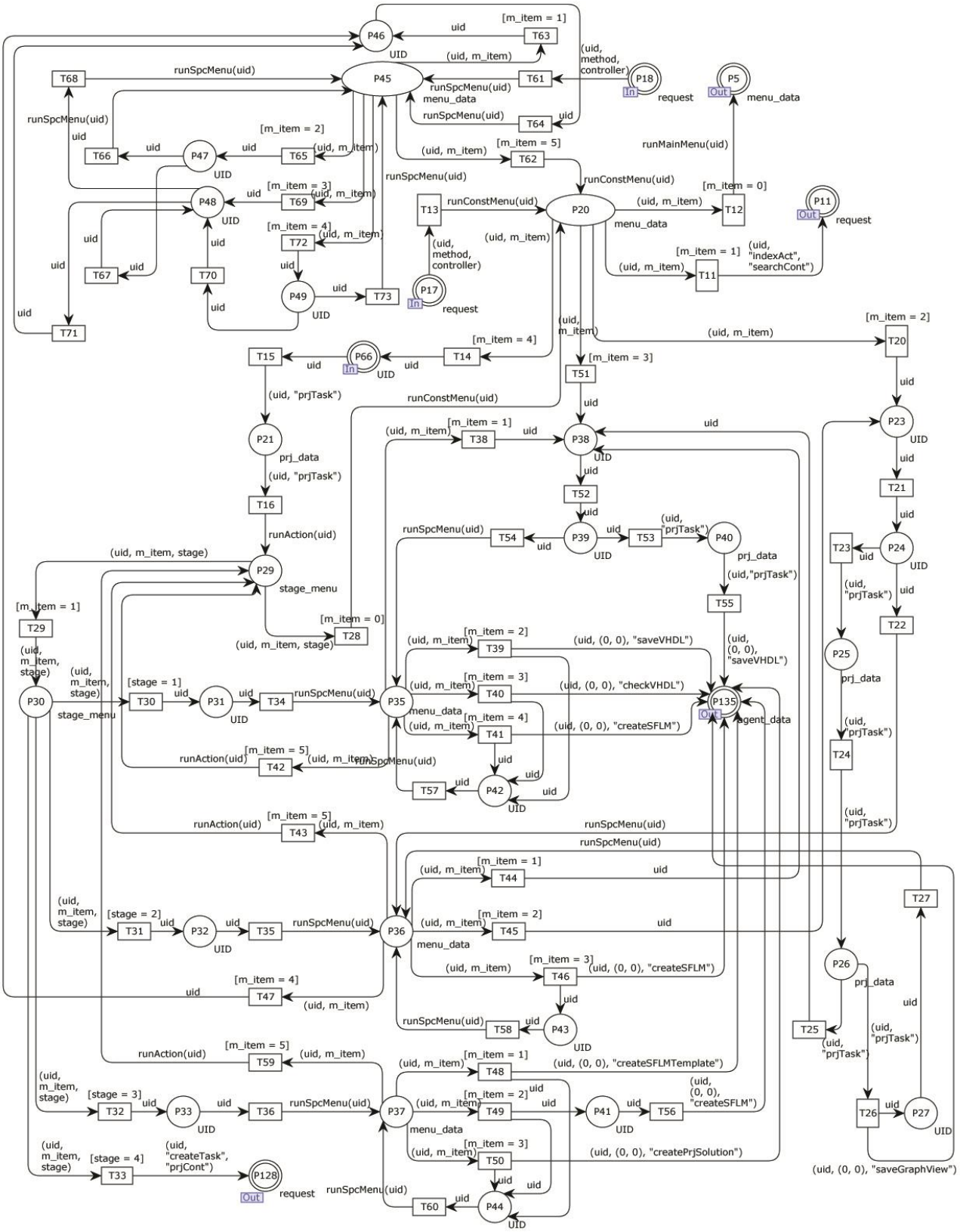


Рис. П4.1. Модель подсистемы конструктора проектного решения (SCPS) на базе сети Петри

Таблица П4.1. Описание позиций и переходов модели подсистемы SCPS

Обозначение элемента	Описание
P5	Переход выполнен
P18	Осуществлен переход к управлению проектными решениями
P11	Переход выполнен
P17	Осуществлен переход на страницу конструктора проектного решения
P66	Осуществлен переход
P21	Результат выбора проектной задачи
P29	Выбрано действие
P38	Осуществлен переход в редактор
P39	Процесс создания VHDL кода завершен
P40	Проектная задача выбрана
P35	Выбрано действие
P42	Продолжить работу в системе
P135	Запрос отправлен
P31	Переход выполнен
P30	Выбрана стадия проектирования
P32	Переход выполнен
P36	Выбрано действие
P37	Выбрано действие
P33	Переход выполнен
P128	Осуществлен переход на страницу создания проектной задачи
P44	Продолжить работу в системе
P41	Процесс генерирования СФМ
P43	Продолжить работу в системе
P47	Выполнен переход
P48	Открыта форма с данными по проектному решению
P46	Обработка завершена
P49	Выбрано проектное решение
P20	Выбран элемент меню
P23	Осуществлен переход в редактор
P24	Процесс создания графической схемы завершен
P25	Проектная задача выбрана
P26	Выбрано действие
P27	Продолжить работу в системе
P45	Выбрано действие
T63	Обработка функциональных блоков СФМ. Внесение изменений
T61	Выбор действия
T64	Открытие формы с измененными данными. Выбор действия
T62	Выбор элемента меню
T13	Выбор элемента меню

T65	Просмотр проектных решении использованных для создания нового решения
T69	Просмотр проектного решения (параметры, функциональные блоки и т.д)
T72	Просмотр найденных проектных решений
T70	Переход к форме с данными
T67	Выбор проектного решения
T66	Выбор действия
T68	Выбор действия
T71	Редактирование выбранной СФЛМ
T14	Переход к созданию/редактированию проектного решения
T15	Выбор проектной задачи
T16	Выбор действия
T73	Выбор действия
T29	Просмотр и выбор доступной стадии разработки проектного решения
T30	Стадия создания проектной задачи завершена. Переход к стадии написания VHDL кода
T34	Выбор действия
T28	Выбор элемента меню
T38	Переход к созданию/ редактированию VHDL кода в редакторе
T51	Переход в редактор с подсветкой синтаксиса VHDL кода
T52	Завершение работы в редакторе
T53	Выбор проектной задачи
T54	Завершение работы в редакторе
T42	Выбор действия
T43	Выбор действия
T39	Переход к сохранению VHDL кода
T40	Переход к проверке VHDL кода
T41	Переход к созданию СФЛМ из VHDL кода
T57	Выбор действия
T31	Стадия написания VHDL кода завершена. Переход к стадии создания СФЛМ из VHDL кода
T35	Выбор действия
T47	Переход к редактированию СФМ
T59	Выбор действия
T36	Выбор действия
T32	Стадия создания СФЛМ из VHDL завершена. Переход к стадии создания шаблона СФЛМ
T33	Переход к стадии создания проектной задачи
T44	Генерирование VHDL кода из СФЛМ. Переход в редактор.
T45	Переход к генерированию графической схемы
T46	Переход к созданию СФЛМ

T58	Выбор действия
T48	Переход к созданию шаблона СФЛМ
T49	Генерирование СФЛМ с новыми параметрами
T50	Завершение создания проектного решения
T60	Выбор действия
T56	Ввод параметров
T12	Переход к предыдущему меню
T11	Переход к форме поиск проектного решения не выходя из конструктора
T20	Переход в редактор графического представления проектного решения
T21	Завершение работы в редакторе
T23	Выбор проектной задачи
T22	Проектная задача выбрана. Выбор действия
T55	Сохранение VHDL кода с привязкой к задаче
T24	Выбор действия
T27	Выбор действия
T25	Генерирование VHDL кода
T26	Сохранение графического представления. Прикрепление его к задаче

Приложение 5. МОДЕЛЬ ПОДСИСТЕМЫ УПРАВЛЕНИЯ ПРОЕКТНЫМИ ЗАДАЧАМИ (SMPT) НА БАЗЕ СЕТИ ПЕТРИ

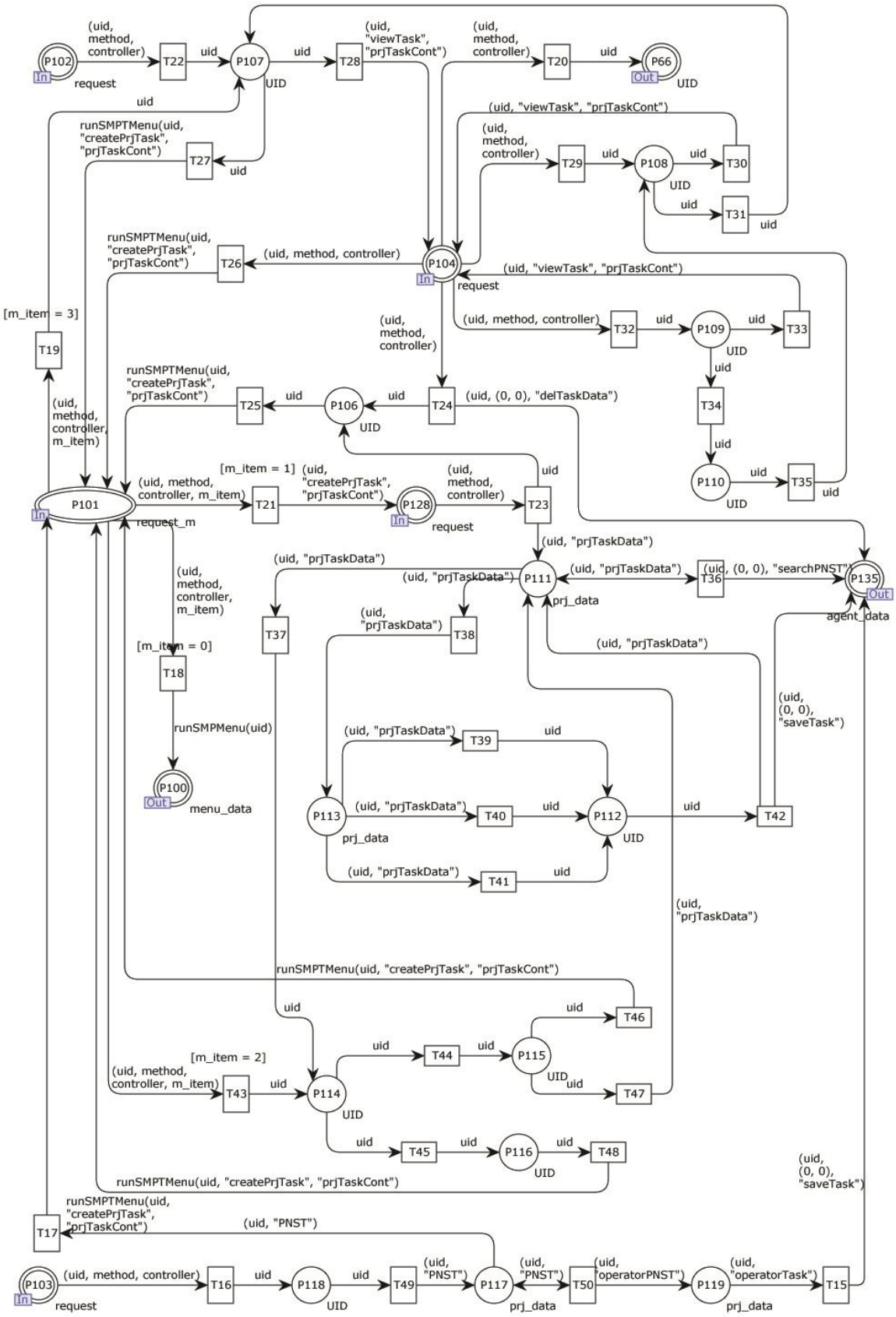


Рис. П5.1. Модель подсистемы управления проектными задачами (SMPT) на базе сети Петри

Таблица П5.1. Описание позиций и переходов модели подсистемы SMPT

Обозначение элемента	Описание
P102	Осуществлен переход на страницу со списком проектных задач
P107	Сформирован список проектных задач
P66	Переход к созданию / редактированию проектного решения выполнен
P101	Выбран элемент меню
P106	Продолжить работу в системе
P128	Осуществлен переход на страницу создания задачи
P111	Данные введены
P113	Осуществлен переход к формированию оператора ПССЗ, описывающего задачу
P100	Переход выполнен
P112	Формирование оператора завершено
P104	Выбрано действие
P108	Осуществлен переход на страницу с информацией по задаче
P109	Осуществлен переход на страницу с данными задачи
P110	Данные отредактированы
P135	Запрос отправлен
P115	Результат выбора
P114	Осуществлен переход на страницу назначения/смены проектировщика
P116	Продолжить работу в системе
P103	Осуществлен переход на страницу с результатом поиска ПССЗ для задачи
P118	Сформирована новая ПССЗ
P117	Процесс обработки операторов ПССЗ
P119	Задача сформирована
T19	Применение фильтра. Вывод списка проектных задач
T22	Применение фильтра
T27	Информирование об отсутствии проектных задач. Выбор элемента меню.
T26	Выбор элемента меню
T25	Выбор элемента меню
T24	Формирование запроса на удаление проектной задачи
T21	Создание проектной задачи
T18	Переход к предыдущему меню
T37	Выбор проектировщика
T38	Формирование оператора ПССЗ
T23	Ввод данных задачи
T36	Формирование запроса на поиск ПССЗ для подобной задачи
T39	Формирование оператора описывающего задачу

T40	Формирование оператора синтеза. Выбор предшествующих задач-операторов
T41	Формирование оператора декомпозиции. Выбор последующих задач-операторов
T42	Формирование запроса (Сохранение задачи)
T17	Завершение формирования списка. Выбор элемента меню
T43	Назначение проектировщика
T16	Формирование ПССЗ из элементов найденной ПССЗ
T49	Декомпозиция задачи по сформированной ПССЗ
T50	Получение оператора ПССЗ И формирование задачи из оператора
T15	Сохранение задачи, формирование запроса и переход к следующему оператору
T44	Поиск вручную. Выбор проектировщика из списка доступных в системе
T45	Запрос на поиск подходящего проектировщика для задачи
T46	Добавление проектировщика к задаче
T47	Выбор элемента меню
T48	Выбор элемента меню
T28	Выбор задачи. Выбор действия с проектной задачей
T29	Просмотр данных
T20	Переход к созданию / редактированию проектного решения
T30	Выбор элемента меню
T31	Переход к списку задач
T32	Редактирование данных
T33	Выбор элемента меню
T34	Редактирование данных
T35	Переход к форме

Приложение 6. МОДЕЛЬ ПОДСИСТЕМЫ УВЕДОМЛЕНИЙ (MES) НА БАЗЕ СЕТИ ПЕТРИ

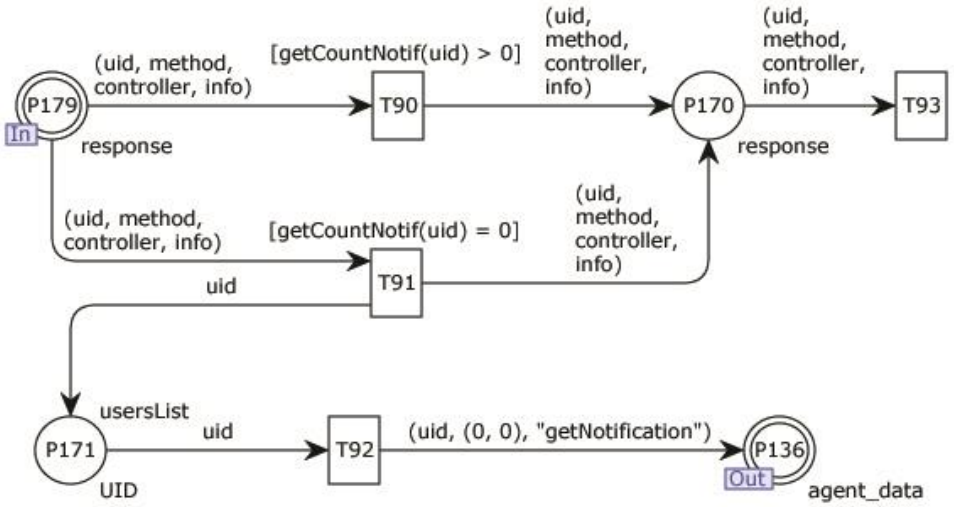


Рис. П6.1. Модель подсистемы уведомлений (MES) на базе сети Петри

Таблица П6.1. Описание позиций и переходов модели подсистемы MES

Обозначение элемента	Описание
P179	Переданы данные по уведомлениям для пользователя
P136	Передан запрос на получение уведомлений для пользователя
P170	Уведомление получено
P171	Запрос сформирован
T90	Получение подготовленного списка уведомлений для проектировщика (если список не пуст)
T91	Запрос на формирование списка уведомлений (если список пуст)
T92	Передача запроса агенту INA
T93	Передача уведомления для вывода на странице пользователя

Приложение 7. МОДЕЛЬ ИНТЕРФЕЙСНОГО АГЕНТА (INA) НА БАЗЕ СЕТИ ПЕТРИ

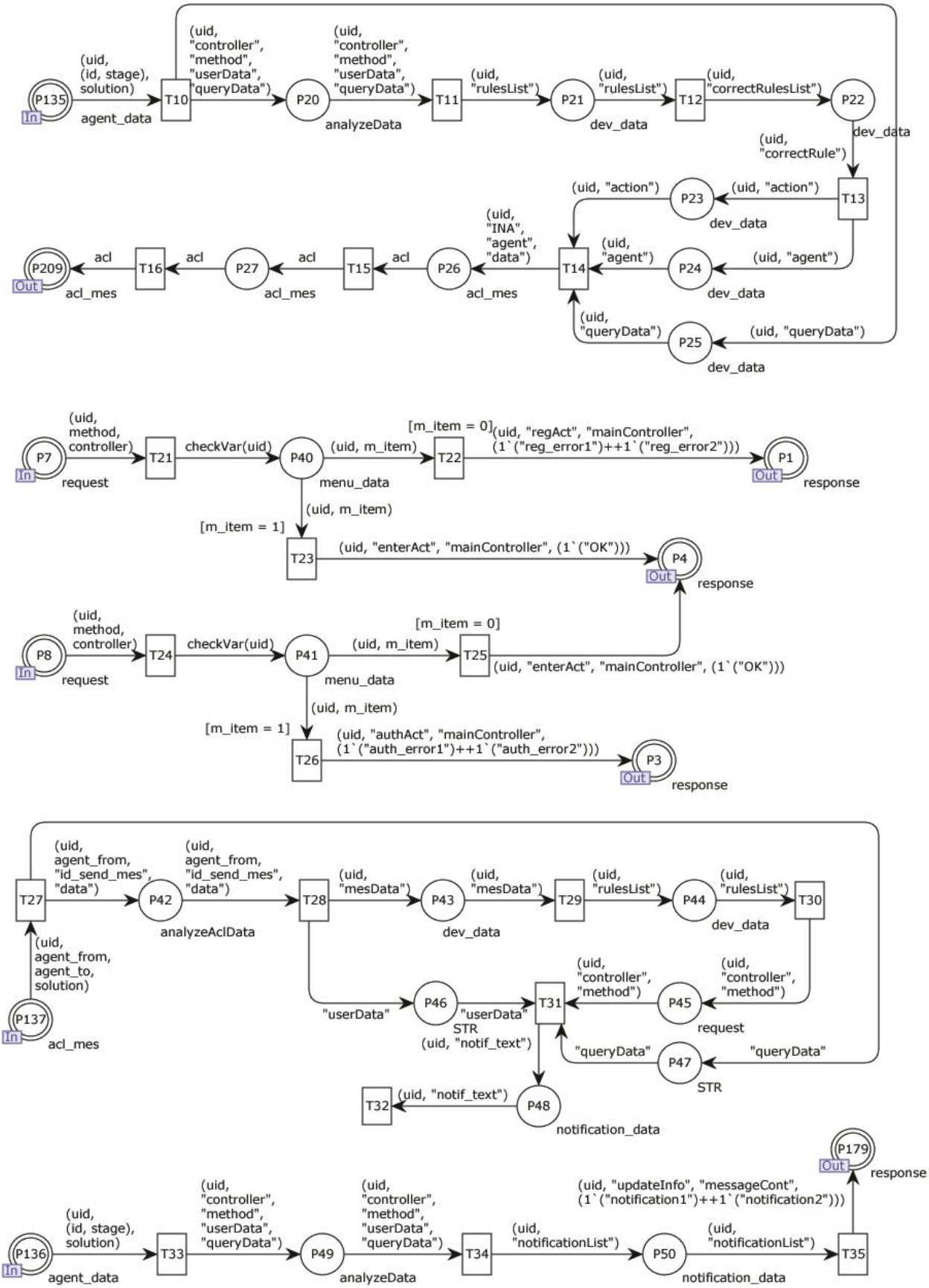


Рис. П7.1. Модель интерфейсного агента (INA) на базе сети Петри

Таблица П7.1. Описание позиций и переходов модели агента INA

Обозначение элемента	Описание
P135	Ожидание запроса от проектировщика
P20	Из запроса выделены необходимые данные
P21	Правила получены
P22	Получен результат поиска
P23	Получено действие
P24	Получен агент-получатель сообщения
P25	Получены данные о проектировщике и запросе
P26	Сообщение сформировано
P27	Связь установлена
P209	Сообщение передано
P7	Ожидание регистрационной информации
P40	Процесс валидации завершен
P1	Сообщение отправлено
P4	Сообщение отправлено
P8	Ожидание данных автризации
P41	Процесс валидации завершен
P3	Сообщение отправлено
P137	Ожидание сообщения от агента
P42	Данные проанализированы
P43	Получены данные отправленного сообщения
P44	Получен результат поиска
P46	Получены данные о проектировщике, запросе, контроллере и методе
P45	Получены данные для отправления (контроллер и метод)
P47	Получены данные по запросу
P48	Уведомление сформировано
P136	Запрос на получение данных по уведомлению
P49	Из запроса выделены необходимые данные
P50	Получен результат поиска
P179	Данные переданы
T10	Анализ запроса. Получение контроллера, метода, данных пользователя и данных запроса.
T11	Получение правил обработки запросов
T12	Поиск подходящего правила работы агента
T13	Анализ правила агента
T14	Формирование сообщения. Сохранение сообщения в базе агента
T15	Установление связи с выбранным агентом
T16	Передача сообщения
T21	Валидация регистрационной информации

T22	Сообщение сведений о некорректной регистрации
T23	Сообщение сведений о регистрации и вход в систему
T24	Валидация авторизационной информации
T25	Сообщение сведений о авторизации и вход в систему
T26	Сообщение сведений о некорректной авторизации
T27	Анализ сообщения. Получение данных агента-отправителя, ID отправляемого сообщения и данных по запросу
T28	Поиск в базе отправляемого сообщения
T29	Поиск подходящего правила работы агента
T30	Анализ правила агента
T31	Формирование уведомления
T32	Сохранение уведомления для пользователя. Перемещение сообщения в историю
T33	Анализ запроса. Получение контроллера, метода, данных пользователя
T34	Поиск уведомления в базе агента по ID проектировщика, контроллеру и методу
T35	Передача данных уведомления в метод контроллера

Приложение 8. МОДЕЛЬ АГЕНТА МАРШРУТИЗАЦИИ (ROA) НА БАЗЕ СЕТИ ПЕТРИ

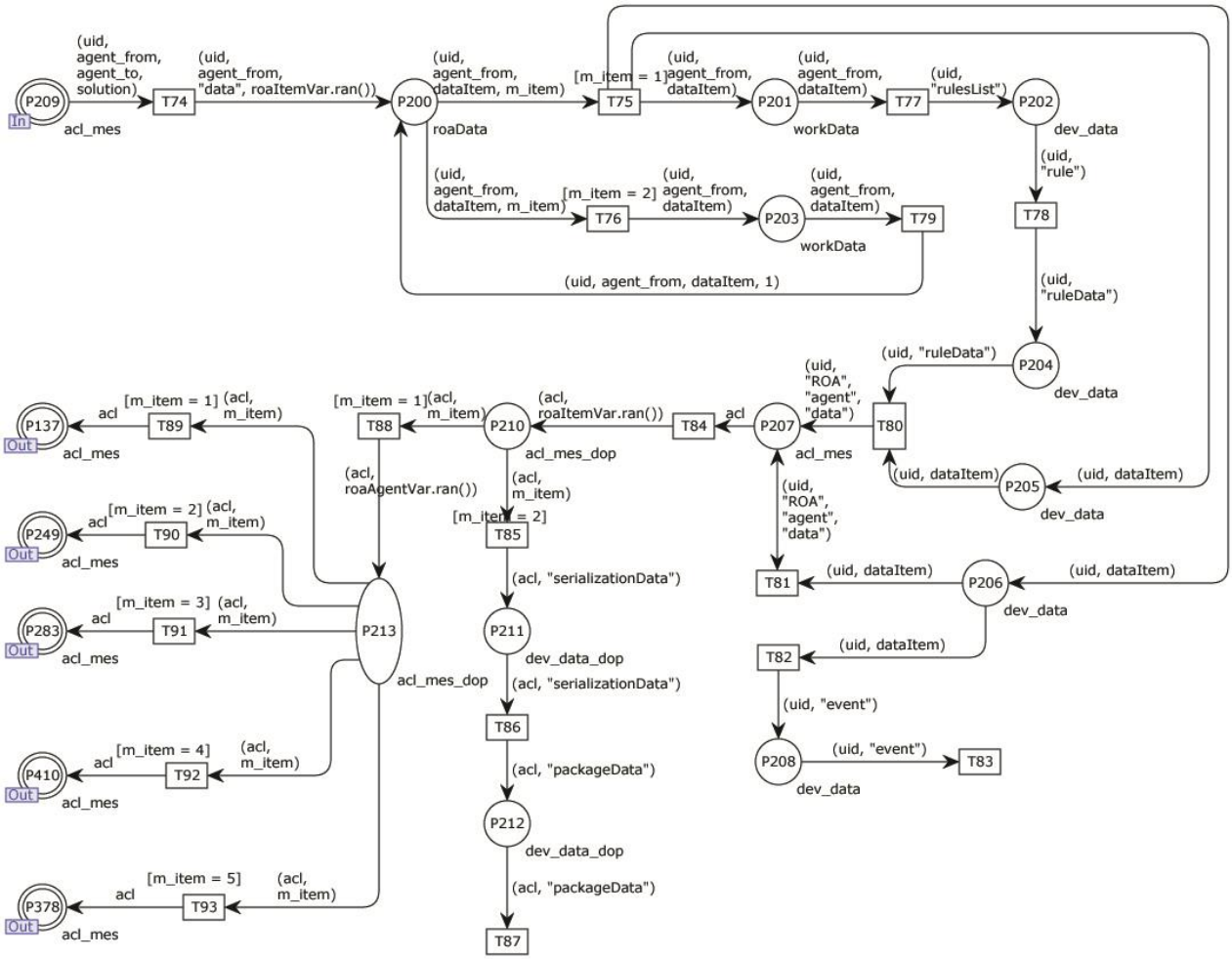


Рис. П8.1. Модель агента маршрутизации (ROA) на базе сети Петри

Таблица П8.1. Описание позиций и переходов модели агента ROA

Обозначение элемента	Описание
P209	Ожидание сообщения от агента
P200	Определена необходимость дополнительной обработки
P201	Данные получены
P202	Получен результат поиска
P203	Демаршаллинг проведен
P204	Получены данные для агента(ов) получателя
P205	Получены данные по проектировщику и запросу
P206	Получен тип события, ID сообщения
P207	Сообщение сформировано
P208	Сообщение найдено
P210	Определена необходимость удаленной передачи сообщения
P211	Проведена сериализация
P212	Маршаллинг проведен

P213	Агент-получатель выбран
P137	Передано сообщение от агента ROA. Сообщение получено
P249	Передано сообщение от агента ROA. Сообщение получено
P283	Передано сообщение от агента ROA. Сообщение получено
P410	Передано сообщение от агента ROA. Сообщение получено
P378	Передано сообщение от агента ROA. Сообщение получено
T74	Анализ сообщения
T75	Получение данных агента-отправителя, ID отправляемого сообщения и данных о проектировщике и запросе
T76	Демаршаллинг
T77	Запуск поиска правила работы агента
T79	Десериализация
T78	Анализ правила агента
T80	Формирование сообщения
T81	Обработка события “запрос”. Сохранение сообщения в базе агента
T82	Обработка события “результат”. Поиск сохраненного события
T83	Перемещение сообщения в историю
T84	Анализ типа события
T85	Сериализация
T86	Маршаллинг
T87	Связь установлена. Передача сообщения удаленному агенту
T88	Выбор агента получателя
T89	Передача сообщения агенту INA
T90	Передача сообщения агенту MAPT
T91	Передача сообщения агенту SEA
T92	Передача сообщения агенту AWM
T93	Передача сообщения агенту ADPS

Приложение 9. МОДЕЛЬ ПОИСКОВОГО АГЕНТА (SEA) НА БАЗЕ СЕТИ ПЕТРИ

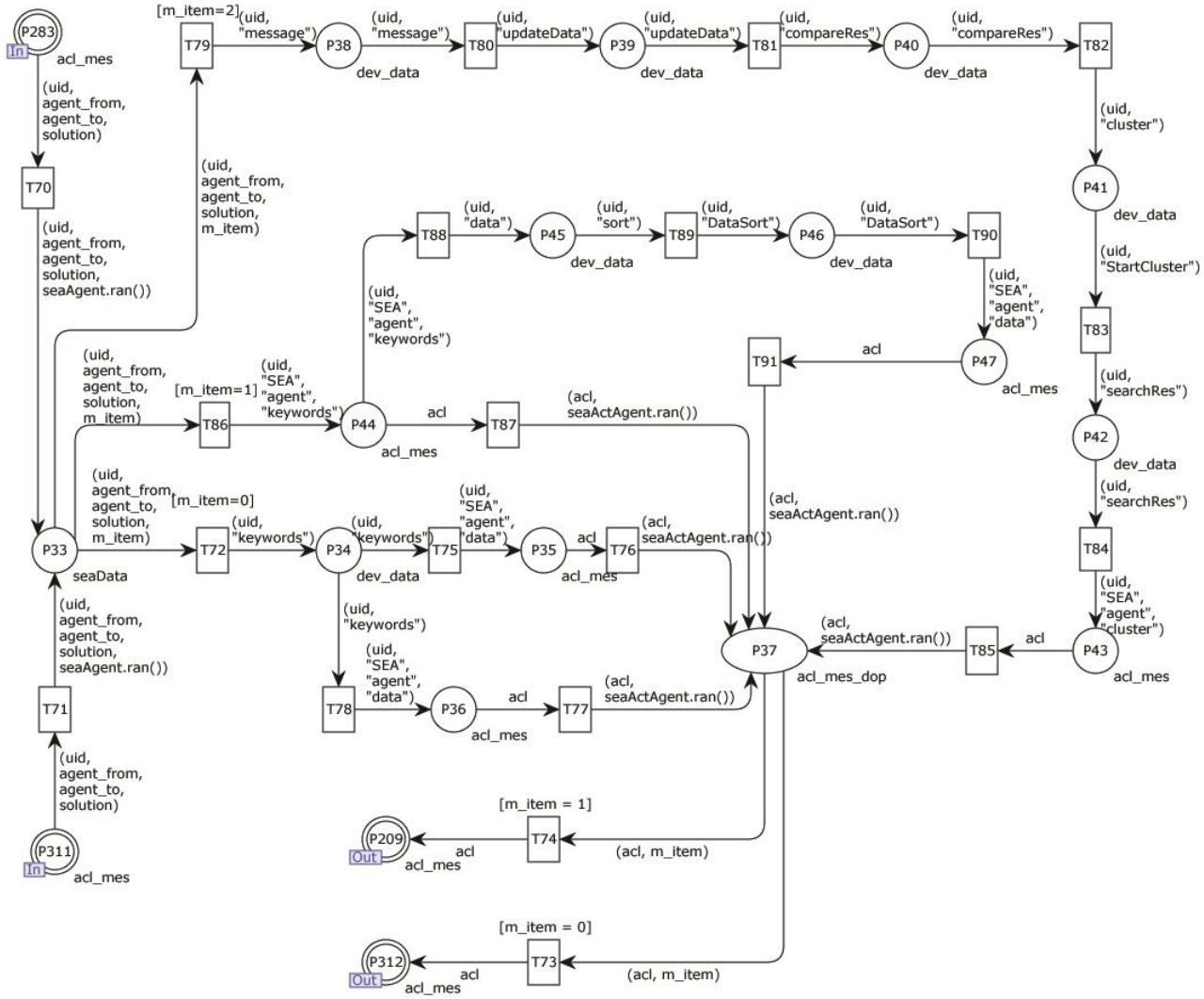


Рис. П9.1. Модель поискового агента (SEA) на базе сети Петри

Таблица П9.1. Описание позиций и переходов модели агента SEA

Обозначение элемента	Описание
P283	Получено сообщение от агента ROA
P311	Получено сообщение от агента AKB
P33	Анализ проведен. Получены данные сообщения
P38	Сообщение получено
P44	Список сформирован
P34	Выделены ключевые слова
P36	Получен результат поиска
P35	Получен запрос на распределенный поиск
P209	Сообщение передано агенту ROA
P45	Получены наборы релевантных данных
P46	Получены отсортированные данные

P47	Данные сохранены
P37	Сообщение сформировано
P312	Сформировано и передано сообщение агенту АКВ
P39	Обновление данных выполнено
P40	Получен результат сравнения
P41	Запущен процесс кластеризации данных
P42	Получен результат поиска
P43	Кластер сформирован
T70	Анализ сообщения от агента ROA. Получение данных агента-отправителя, ID отправляемого сообщения и данных по запросу
T71	Анализ сообщения от агента АКВ. Получение данных агента-отправителя, ID отправляемого сообщения и данных по запросу
T72	Выделение в поисковом запросе ключевых слов и доп. параметров (при расширенном поиске)
T86	Формирование списка полученных данных
T88	Обработка данных с учетом значимых слов Выделение части СФЛМ наиболее относящееся к запрос
T87	Выбор агента маршрутизации. Информирование об отсутствии результатов поиска
T76	Выбор агента маршрутизации. Формирование сообщения
T77	Выбор интерфейсного агента. Формирование сообщения
T74	Передача сообщения агенту ROA
T73	Передача сообщения агенту АКВ
T78	Запуск поиска данных в базе агента
T79	Поиск отправленного сообщения, соответствующего полученным результатам
T80	Обновление данных (время запроса и результат запроса) в сообщении
T81	Сравнение времени поиска с параметром оптимального времени. Логирование событий
T89	Сортировка данных учитывая специфику деятельности проектировщика и часто посещаемые источники поиска СФЛМ
T90	Сохранение найденных данных в базе агента
T91	Выбор агента маршрутизации. Формирование сообщения
T82	Не оптимальное время поиска. Запуск процесса кластеризации данных
T83	Поиск в базе агента, сообщения с подобными ключевыми словами и доп. параметрами поиска
T84	Выделение однотипных параметров. Формирование кластера
T85	Выбор агента базы данных. Формирование сообщения

Приложение 10. МОДЕЛЬ АГЕНТА СИНТЕЗА ПРОЕКТНЫХ РЕШЕНИЙ (ASPS) НА БАЗЕ СЕТИ ПЕТРИ

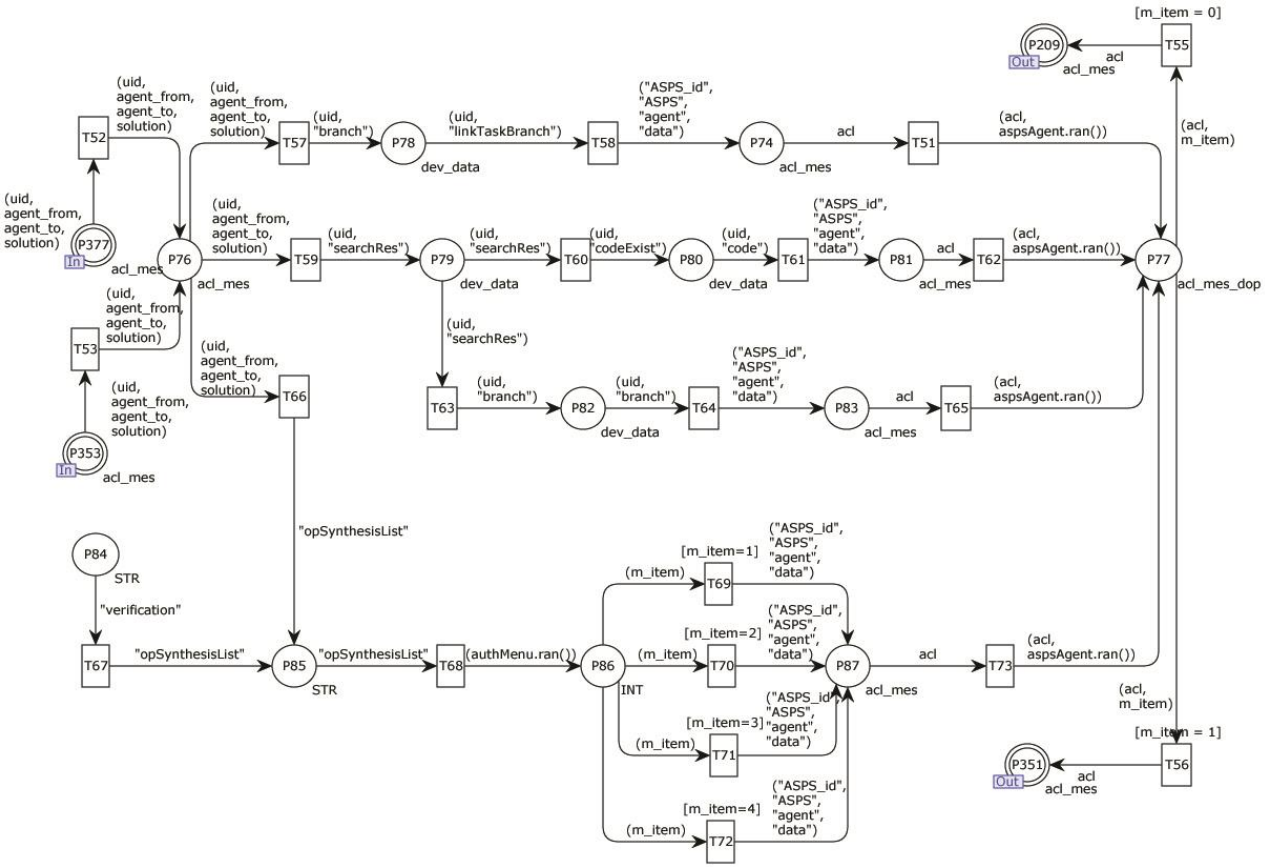


Рис. П10.1. Модель агента синтеза проектных решений (ASPS) на базе сети Петри

Таблица П10.1. Описание позиций и переходов модели агента ASPS

Обозначение элемента	Описание
P377	Передано сообщение от агента ADPS. Сообщение получено
P76	Анализ проведен. Данные получены
P353	Передано сообщение от агента AKB. Сообщение получено
P78	Создана ветка разработки
P74	Связь между задачей и веткой разработки установлена
P79	Получен результат поиска ветки разработки
P80	Код был залит ранее
P81	Получен результат слияния версий
P82	Создана ветка разработки
P83	Код успешно залит в ветку разработки
P77	Сообщение сформировано
P209	Сообщение передано агенту ROA
P84	Периодическая проверка графа ПССЗ на наличие завершенных стадий операторов (проектных задач), которые предшествуют оператору синтеза в графе ПССЗ

P85	Процесс анализа завершен
P86	Получена общая стадия, на основе которой будет производится синтез
P87	Получено новое проектное решение. Установлена связь с оператором синтеза
P351	Сообщение передано агенту АКВ
T52	Анализ сообщения от агента ADPS. Получение данных агента-отправителя, ID отправляемого сообщения и данных по запросу.
T53	Анализ сообщения от агента АКВ. Получение данных агента-отправителя, ID отправляемого сообщения и данных по запросу.
T57	Информирование о начале работы над задачей. Создание ветки разработки в Git
T58	Установление связи ветки разработки с задачей
T51	Формирование сообщения (создание ветки разработки). Выбор агента маршрутизации
T59	Получен VHDL Код. Поиск ветки разработки для задачи
T60	Внесение кода в ветку разработки
T61	Обновление кода. Слияние версий (локальной и серверной)
T62	Формирование сообщения (результат слияния, разрешение конфликтов). Выбор агента маршрутизации.
T63	Создание ветки разработки Git
T64	Внесение кода в ветку разработки
T65	Формирование сообщения (уведомление о интеграции кода). Выбор агента маршрутизации
T66	Завершена стадия разработки проектной задачи. Выбор оператора синтеза и список предшествующих операторов среди которых есть оператор сменивший стадию
T67	Запуск проверки графа ПССЗ. Поиск операторов синтеза (проектных задач) в графе ПССЗ (проекте)
T68	Получение общей стадии разработки для предшествующих операторов
T69	Синтез VHDL кода. Объединение веток разработки в Git
T70	Синтез СФЛМ
T71	Синтез шаблонов СФЛМ
T72	Синтез графических блоков
T73	Формирование сообщения (сохранение данных). Выбор агента маршрутизации
T55	Передача сообщения агенту ROA
T56	Передача сообщения агенту АКВ

Приложение 11. МОДЕЛЬ АГЕНТА РАЗРАБОТКИ ПРОЕКТНОГО РЕШЕНИЯ (ADPS) НА БАЗЕ СЕТИ ПЕТРИ

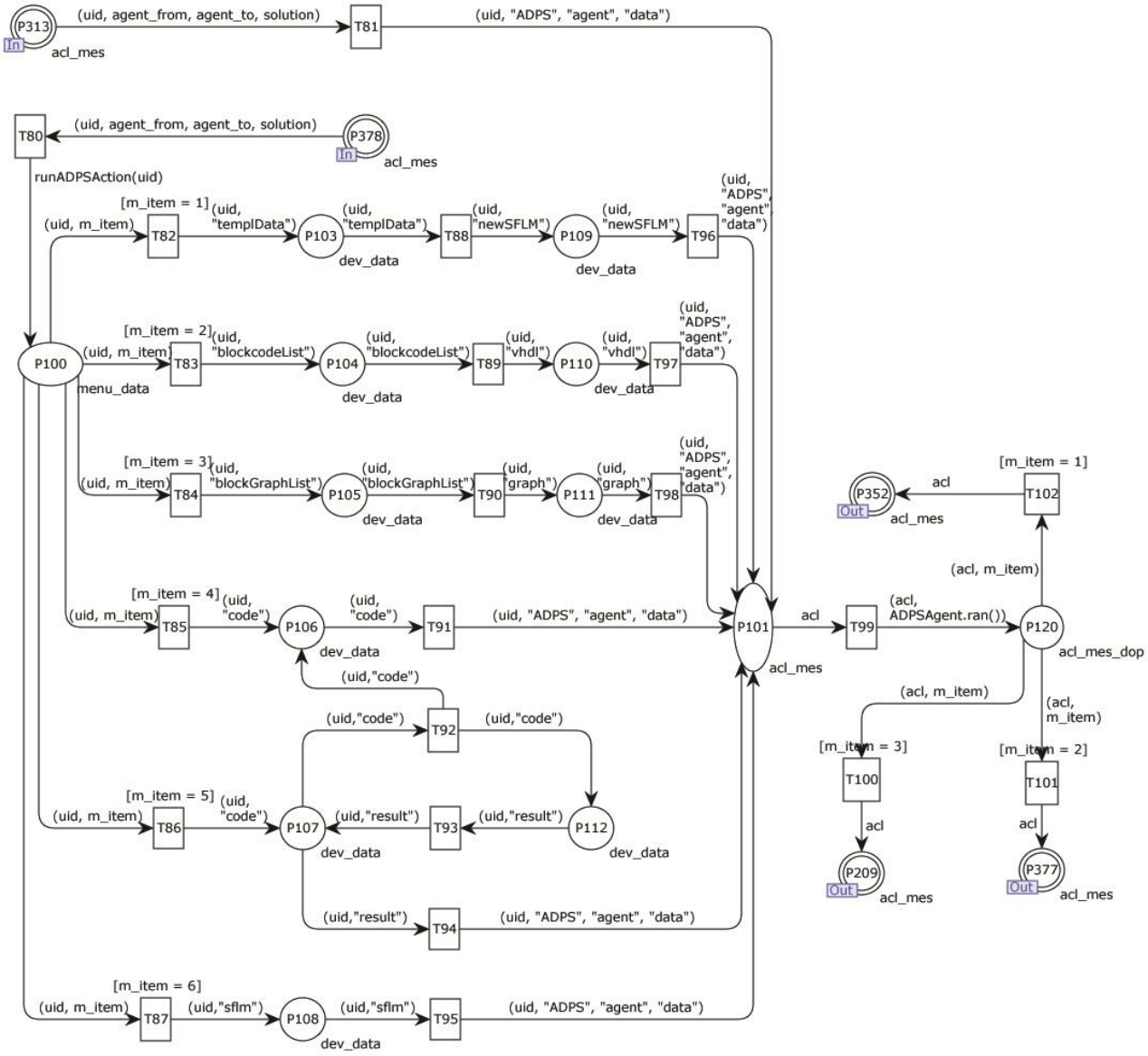


Рис. П11.1. Модель агента разработки проектного решения (ADPS) на базе сети Петри

Таблица П11.1. Описание позиций и переходов модели агента ADPS

Обозначение элемента	Описание
P100	Анализ проведен и данные сообщения получены
P313	Передано сообщение от агента AKB. Сообщение получено
P378	Передано сообщение от агента ROA. Сообщение получено
P103	Проверка данных шаблона завершена
P109	Сформирован СФЛМ с новыми параметрами
P104	Получены блоки СФЛМ в виде VHDL кода
P110	Процесс формирования VHDL кода завершен
P105	Получены блоки СФЛМ в виде графических блоков

P111	Процесс формирования графической схемы завершен
P106	Проверка кода завершена
P107	Получен результат
P112	Запущен анализ кода
P108	СФЛМ сформирована
P101	Сообщение сформировано
P120	Агент получатель выбран
P352	Передано сообщение от агента ADPS. Сообщение получено
P209	Сформировано и передано сообщение агенту ROA
P377	Передано сообщение от агента ADPS. Сообщение получено
T80	Анализ сообщения. Получение данных агента-отправителя, ID отправляемого сообщения и данных по запросу
T81	Формирование сообщения (результат запроса к агенту AKB)
T82	Создание новой СФЛМ из шаблона СФЛМ
T83	Формирование VHDL из СФЛМ. Обрабатываем функциональные блоки СФЛМ
T84	Формирования графического представления из СФЛМ
T85	Лексический и семантический анализ VHDL кода
T86	Создание/ редактирования VHDL кода
T87	Формирование СФЛМ из VHDL кода
T88	Расчет параметров на основе начальных данных и подстановка в шаблон СФЛМ
T96	Формирование сообщения (СФЛМ сформировано). Выбор агента маршрутизации
T89	Объединение блоков VHDL кода
T97	Формируем сообщение (результат формирования VHDL кода). Выбираем агента маршрутизации
T90	Объединение графических блоков
T98	Формируем сообщение (результат формирования графической схемы). Выбираем агента маршрутизации
T91	Формирование сообщения (результат анализа кода) Выбор агента маршрутизации
T92	Лексический и семантический анализ VHDL кода
T93	Продолжение выполнения операции
T94	Формирование сообщения (сохранения VHDL кода). Выбор агента базы данных
T95	Формирование сообщения (сохранение СФМ). Выбор агента базы знаний
T99	Связь установлена. Передача сообщения
T102	Передача сообщения агенту AKB
T100	Передача сообщения агенту ROA
T101	Передача сообщения агенту ASPS

Приложение 12. МОДЕЛЬ АГЕНТА УПРАВЛЕНИЯ ПРОЕКТНЫМИ ЗАДАЧАМИ (MAPT) НА БАЗЕ СЕТИ ПЕТРИ

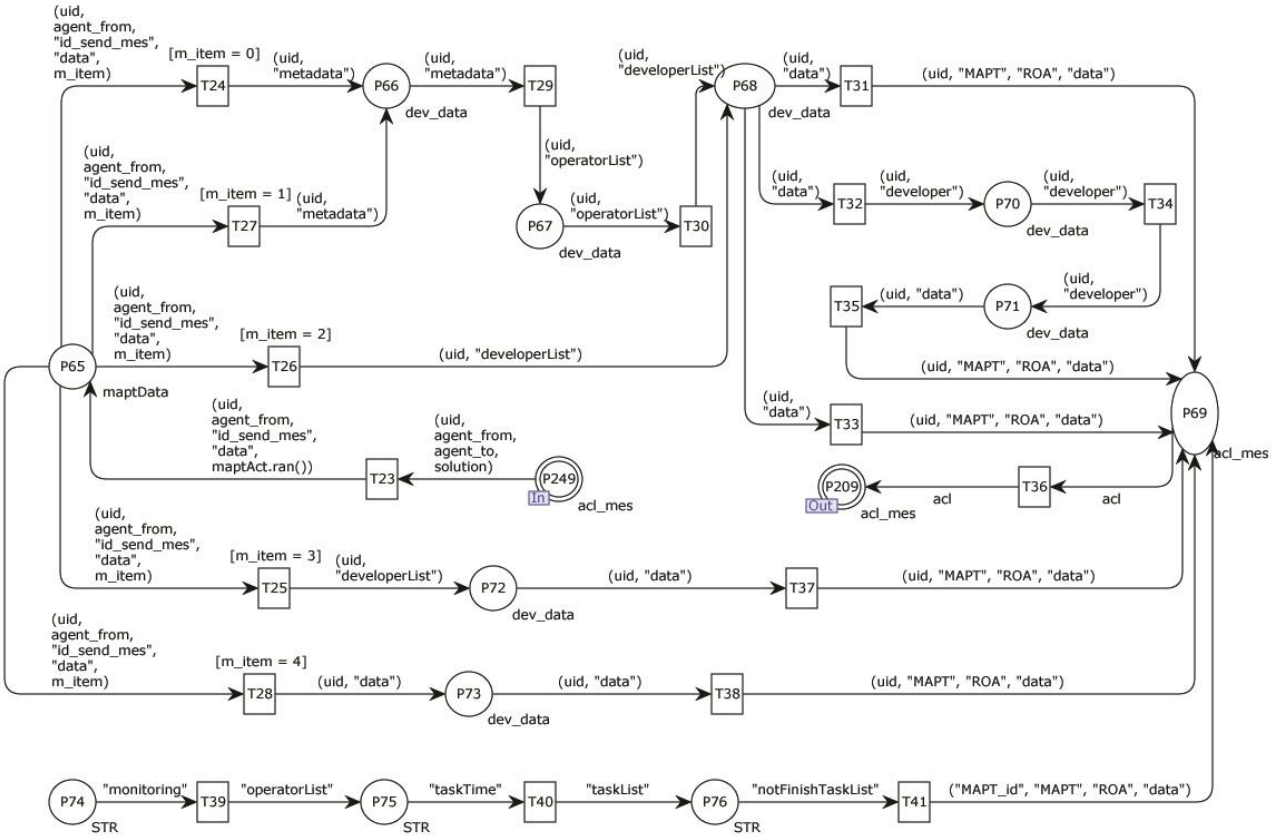


Рис. П12.1. Модель агента управления проектными задачами (MAPT) на базе сети Петри

Таблица П12.1. Описание позиций и переходов модели агента MAPT

Обозначение элемента	Описание
P65	Данные сообщения обработаны
P66	Получены ключевые слова (метаданные)
P67	Сформирован иерархический список операторов
P68	Список потенциальных проектировщиков сформирован
P70	Проектировщик выбран
P71	Команда проектировщиков обновлена
P69	Сообщение сформировано
P209	Сообщение передано
P249	Получено сообщение от агента ROA
P72	Получен упорядоченный список
P73	Данные успешно обновлены
P74	Мониторинг процесса проектирования
P75	Выбраны операторы ПССЗ

P76	Получен список незавершенных задач, с окончанным/заканчивающимся сроком выполнения
T24	Поиск подобной проектной задачи и проектировщика для его выполнения
T27	Декомпозиция проектной задачи
T26	Назначение задачи проектировщику
T23	Анализ сообщения. Получение данных агента-отправителя, ID отправляемого сообщения и данных по запросу
T25	Получение списка задач проектировщика с указанием времени выполнения. Сортировка задач по приоритету и времени выполнения
T28	Редактирование проектной задачи
T39	Просмотр ПССЗ на наличие операторов и задач по которым ведется разработка
T29	Поиск по метаданным среди операторов ПССЗ содержащих похожую задачу. Сортировка по релевантности
T30	Добавление проектировщиков в список со временем возможного завершения задачи (время до завершения задач + время решения текущей задачи). Сортировка списка
T31	Формируем сообщение (в ПССЗ найдены подобные задачи и проектировщики способные решить данную задачу). Выбираем агента маршрутизации
T32	Выбор первого из списка возможных проектировщиков в качестве ответственного по задаче
T33	Назначение проектировщика каждой найденной задаче и подзадаче. Формируем сообщение (новая ПССЗ, описывающая декомпозицию). Выбираем агента маршрутизации
T35	Формируем сообщение (уведомление о назначении проектировщика). Выбираем агента маршрутизации
T37	Формируем сообщение. (Передача задач проектировщика). Выбираем агента маршрутизации
T38	Формируем сообщение (Обновление графика работ, Уведомление). Выбираем агента маршрутизации
T34	Добавления проектировщика к команде, в случае его отсутствия
T36	Передача сообщения
T41	Формирование уведомлений по выяснению причины не завершения задачи или информировании об окончании срока разработки. Выбираем агента маршрутизации
T40	Анализ срока выполнения задач

Приложение 13. МОДЕЛЬ АГЕНТА БАЗЫ ДАННЫХ (AKB) НА БАЗЕ СЕТИ ПЕТРИ

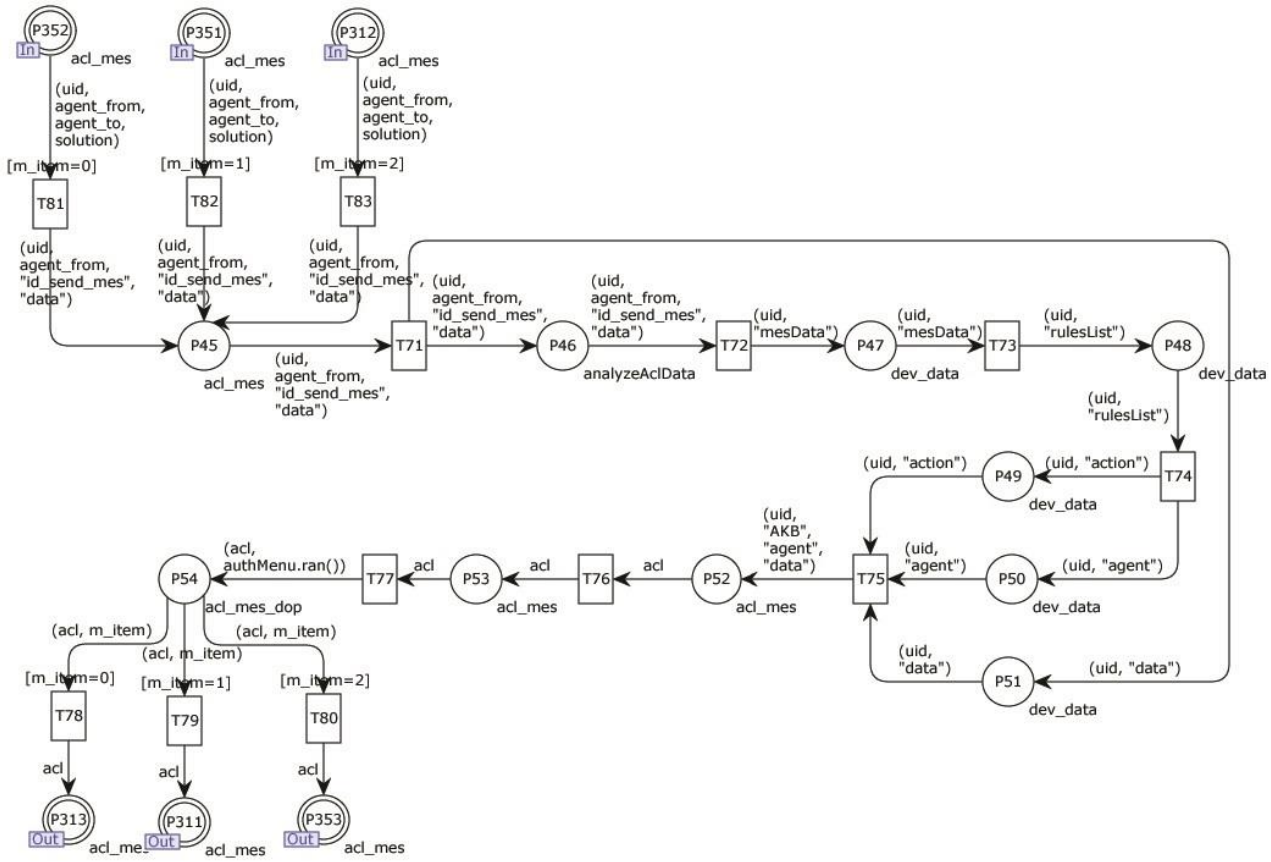


Рис. П13.1. Модель агента базы данных (AKB) на базе сети Петри

Таблица П13.1. Описание позиций и переходов модели агента AKB

Обозначение элемента	Описание
P352	Ожидание сообщения от агента ADPS
P351	Ожидание сообщения от агента ASPS
P312	Ожидание сообщения от агента SEA
P45	Сообщение готово к обработке
P46	Переданные данные проанализированы
P47	Получены данные отправленного сообщения
P48	Получен результат поиска правила функционирования
P49	Получены данные для отправления (агент получатель, данные для обработки)
P50	Получены данные о проектировщике, агенте отправителе и само содержимое
P51	Получены данные из сообщения агента
P52	Сообщение сформировано
P53	Связь установлена
P54	Агент получатель выбран

P313	Сообщение передано агенту ADPS
P311	Сообщение передано агенту SEA
P353	Сообщение передано агенту ASPS
T81	Передача сообщения от агента ADPS на обработку
T82	Передача сообщения от агента ASPS на обработку
T83	Передача сообщения от агента SEA на обработку
T71	Анализ сообщения. Получение данных агента-отправителя, ID отправляемого сообщения и данных по запросу
T72	Поиск в базе отправляемого сообщения
T73	Поиск подходящего правила работы агента
T74	Анализ правила агента
T75	Формирование сообщения. Сохранение сообщения в базе агента
T76	Установление связи с выбранным агентом
T77	Выбор агента для передачи сообщения
T78	Передача сообщения агенту ADPS
T79	Передача сообщения агенту SEA
T80	Передача сообщения агенту ASPS

Приложение 14. МОДЕЛЬ АГЕНТА РАБОЧЕЙ ПАМЯТИ (AWM) НА БАЗЕ СЕТИ ПЕТРИ

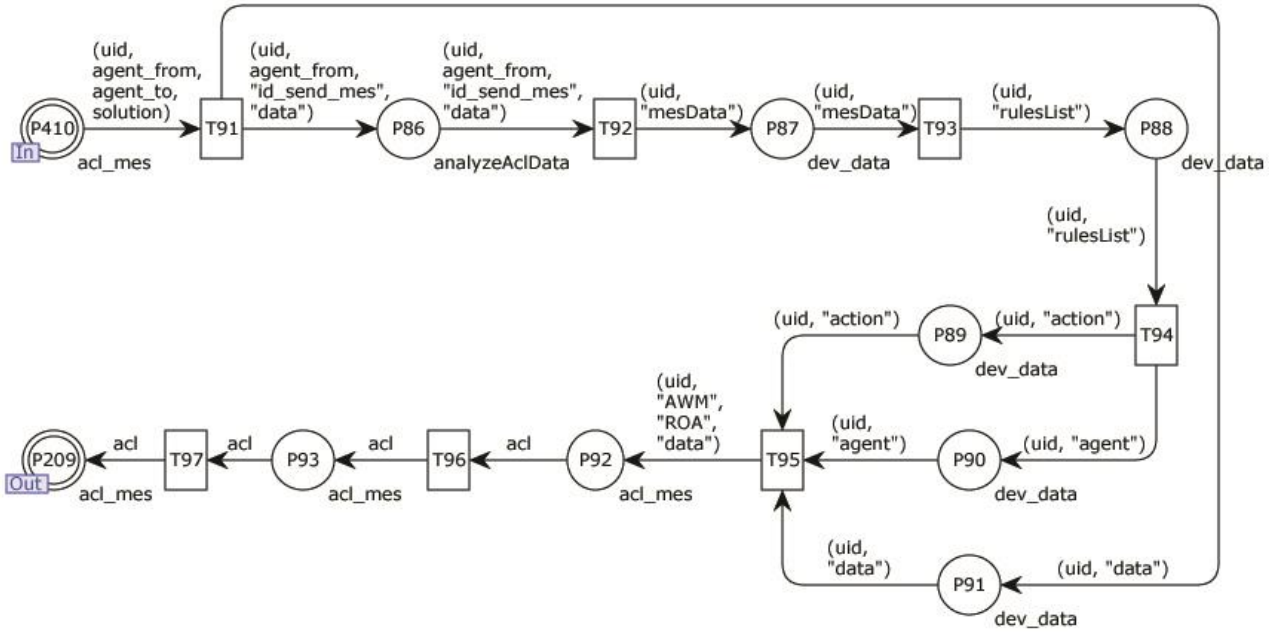


Рис. П14.1. Модель агента рабочей памяти (AWM) на базе сети Петри

Таблица П14.1. Описание позиций и переходов модели агента AWM

Обозначение элемента	Описание
P410	Ожидание сообщения от агента ROA
P86	Переданные данные проанализированы
P87	Получены данные отправленного сообщения
P88	Получен результат поиска правила функционирования
P89	Получены данные для отправления (агент получатель, данные для обработки)
P90	Получены данные о проектировщике, агенте отправителе и само содержимое
P91	Получены данные из сообщения агента
P92	Сообщение сформировано
P93	Связь установлена
P209	Сообщение передано
T91	Анализ сообщения. Получение данных агента-отправителя, ID отправляемого сообщения и данных по запросу
T92	Поиск в базе отправляемого сообщения
T93	Поиск подходящего правила работы агента
T94	Анализ правила агента
T95	Формирование сообщения. Сохранение сообщения в базе агента
T96	Установление связи с выбранным агентом
T97	Передача сообщения

Приложение 15. ВЗАИМОДЕЙСТВИЕ АГЕНТОВ В СИСТЕМЕ СРП

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include "struct.h"

void error(const char *msg)
{
	perror(msg);
	exit(0);
}

int main(int argc, char *argv[])
{
	int sockfd, portno, n;
	struct sockaddr_in serv_addr;
	struct hostent *server;

	char buffer[256];
	if (argc < 3) {
		fprintf(stderr,"usage %s hostname port\n", argv[0]);
		exit(0);
	}
	portno = atoi(argv[2]);
	sockfd = socket(AF_INET, SOCK_STREAM, 0);
	if (sockfd < 0)
	{
		error("ERROR opening socket");
	}
	server = gethostbyname(argv[1]);
	if (server == NULL) {
		fprintf(stderr,"ERROR, no such host\n");
		exit(0);
	}
	bzero((char *) &serv_addr, sizeof(serv_addr));
	serv_addr.sin_family = AF_INET;
	bcopy((char *)server->h_addr,
	(char *)&serv_addr.sin_addr.s_addr,
	server->h_length);
	serv_addr.sin_port = htons(portno);
	if (connect(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0)
	{
		error("ERROR connecting");
	}
	struct ACL *msg;
	unsigned char buffer1[sizeof(msg)];
```

```
	struct ACL theMsg;
	int iResult = recv(sockfd, buffer1, sizeof(msg), 0);
	if ( iResult > 0 )
	{
		msg = (struct ACL *)&buffer1;
		theMsg = *msg;
	}
	close(sockfd);
	return 0;
}
```

Приложение 16. АНАЛИЗ VHDL КОДА И ПОЛУЧЕНИЕ СФЛМ В WEB-ОРИЕНТИРОВАННОМ ТРАНСЛЯТОРЕ

```
<?function analyze()
{
	$next_elem = null; // следующий элемент
	$next_elem_stack = array(); // стек следующих элементов (при рекурсии)
	$possible_options = null; // правило последовательности элементов vhdl кода
	$possible_options_stack = array(); // стек правил последовательности элементов vhdl кода (при рекурсии)
	$current_word = ""; // текущее слово
	$possible_variants = array(); // список возможных вариантов, в случае ошибки в слове
	$possible_variants_stack = array();
	$stack_prev_state = array(); // стек предыдущих состояний (в случае ошибки в коде)
	$vhdl_struct_code = array(); // структура vhdl
	$current_list_delimeter = array(); // список текущих разделителей
	$count_check_word = 0; // количество обработанных слов
	$current_delimeter = ""; // текущий разделитель
	$vhdl_struct = "";

	$error_parsing = array(); // ошибки при анализе

	$start = null;
	$end = null;
	foreach ($this->data_array as $key => $string)
	{
		$string = trim($string);
		if ($current_word != "")
		{
			if (count($current_list_delimeter) == 0)
			{
				array_push($current_list_delimeter, array("VALUE" => " ", "POS" => -1));
			}
			array_push($vhdl_struct_code[$str_num], array("WORD" => $current_word, "POS" => array("START" => $start, "END" => $end), "DELIM" => $current_list_delimeter));
			$current_word = "";
			$start = null;
			$end = null;
			$current_list_delimeter = array();
		}
		$str_num = $key + 1;
		if (($string != "\r\n") || ($string != "\n"))
		{
			$vhdl_struct_code[$str_num] = array();
			$stringLength = iconv_strlen($string);
			for ($i = 0; $i < $stringLength; $i++)
			{
				if (in_array($string[$i], $this->delimeters))
				{
					if (!isset($end) && (isset($start)))
					{
```

```
                                        $end = $i - 1;
                                }
                                if (($string[$i] != " ") && ($string[$i] != "\t"))
                                {
                                        $current_delimeter = $string[$i];
                                        array_push($current_list_delimeter, array("VALUE"
=> $current_delimeter, "POS" => $i));
                                }
                                else if ($string[$i] != "\t")
                                {
                                        array_push($current_list_delimeter, array("VALUE"
=> $string[$i], "POS" => $i));
                                }
                        }
                        else
                        {
                                if (!isset($start))
                                {
                                        $start = $i;
                                }
                                $current_word .= $string[$i];
                                $current_list_delimeter = array();
                        }
                        if (((count($current_list_delimeter) > 0) && (!isset($string[$i + 1]) ||
(!in_array($string[$i + 1], $this->delimeters)))) || ($i == ($stringLength - 1)))
                        {
                                foreach($current_list_delimeter as $delim_key =>
$delim_value)
                                {
                                        if ($delim_value["VALUE"] == " ")
                                        {
                                                unset($current_list_delimeter[$delim_key]);
                                        }
                                }
                                if (count($current_list_delimeter) == 0)
                                {
                                        $current_list_delimeter = array();
                                        array_push($current_list_delimeter, array("VALUE"
=> " ", "POS" => $i));
                                }
                                array_push($vhdl_struct_code[$str_num], array("WORD" =>
$current_word, "POS" => array("START" => $start, "END" => $end), "DELIM" =>
$current_list_delimeter));
                                $current_word = "";
                                $start = null;
                                $end = null;
                                $current_list_delimeter = array();
                        }
                }
        }
}
```

```
	$possible_variants = null;
	for($str_num = 1; $str_num <= count($vhdl_struct_code); $str_num++)
	{
		$struct_code = $vhdl_struct_code[$str_num];
		for($num_word = 0; count($struct_code) > 0; $num_word++)
		{
			$struct_value = $struct_code[$num_word];
			if (isset($struct_value["DELIM"]))
				$current_list_delimeter = $struct_value["DELIM"];
			if (isset($struct_value["WORD"]))
				$current_word = $struct_value["WORD"];
			if (!$this->find_comment($current_list_delimeter, true, $current_word))
			{
				$pos = null;
				if (isset($current_word) && ($current_word != ""))
				{
					$pos = array_search($current_word, $this->path_names);
					if ($pos === false)
					{
						if ((count($possible_options) == 0) &&
(count($possible_options_stack) == 0))
						{
							$this->createError($error_parsing,
"Неопознанная ошибка в слове '".$current_word."'. Ошибка в строке ".$str_num.".", $str_num,
"error");
							$this->errors_list = $error_parsing;
							return false;
						}
						else
						{
							$new_word = "";
							if(preg_match("/^[a-zA-Z][a-zA-Z0-9_]*/",
$current_word))
							{
								$new_word = "var";
							}
							else if(preg_match("/^[0-9]+/", $current_word))
							{
								$new_word = "digits";
							}

							$pos = array_search($new_word, $this-
>path_names);
						}
					}
					if (!isset($pos) || $pos === false)
					{
						unset($struct_value["WORD"]);
						unset($struct_code[$num_word]["WORD"]);
						$this->createError($error_parsing, "Неопознанная
ошибка в слове '".$current_word."'. Ошибка в строке ".$str_num.".", $str_num, "error");
						$this->errors_list = $error_parsing;
```

```
                                        return false;
                                }
                        }
                        else if (isset($current_list_delimeter) &&
(count($current_list_delimeter) > 0))
                        {
                                reset($current_list_delimeter);
                                $current_delimeter = array_shift($current_list_delimeter);
                                array_shift($struct_value["DELIM"]);
                                array_shift($struct_code[$num_word]["DELIM"]);
                                $pos = array_search($current_delimeter["VALUE"], $this-
>path_names);
                                if (!isset($pos) || $pos === false)
                                {
                                        $this->createError($error_parsing, "Неопознанная
ошибка в разделителе '".$current_delimeter."'. Ошибка в строке ".$str_num.".", $str_num,
"error");
                                        $this->errors_list = $error_parsing;
                                        return false;
                                }
                        }

                        if (!$this->get_next_block($error_parsing, $possible_options,
$possible_options_stack, $pos, $current_list_delimeter, $current_word, $vhdl_struct, $str_num,
$num_word, $struct_code, $struct_value, $possible_variants, $current_delimeter,
$vhdl_struct_code, $possible_variants_stack))
                        {
                                $this->createError($error_parsing, "Неопознанная ошибка в
слове '".$current_word."'. Ошибка в строке ".$str_num.".", $str_num, "error");
                                $this->errors_list = $error_parsing;
                                return false;
                        }
                }
                else
                {
                        $this->comment_processing($struct_code, $num_word, $vhdl_struct);
                        unset($struct_value);
                        unset($struct_code[$num_word]);
                        break;
                }

                if (count($struct_value["DELIM"]) == 0)
                {
                        unset($struct_value);
                        unset($struct_code[$num_word]);
                }
        }
}
$this->errors_list = $error_parsing;
$this->vhdl_struct .= $vhdl_struct."</vhdl_programm>";}?>
```

Приложение 17. СВИДЕТЕЛЬСТВО (РОСПАТЕНТ) О ГОСУДАРСТВЕННОЙ РЕГИСТРАЦИИ ПРОГРАММЫ ДЛЯ ЭВМ №2015610356

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2015610356

Web-ориентированный транслятор VHDL описаний в шаблоны структурно-функциональных лингвистических моделей

Правообладатель: ***федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Ульяновский государственный технический университет» (RU)***

Авторы: ***Афанасьев Александр Николаевич (RU),***
Хородов Виталий Сергеевич (RU)

Заявка № **2014661186**
Дата поступления **05 ноября 2014 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***12 января 2015 г.***

Врио руководителя Федеральной службы по интеллектуальной собственности

Л.Л. Кирий

Приложение 18. АКТЫ О ВНЕДРЕНИИ

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное
образовательное учреждение высшего
профессионального образования
**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
(УлГТУ)**
Северный венец ул., д. 32,
г. Ульяновск, 432027, Россия
Тел.: (8422) 43-06-43; факс: (8422) 43-02-37
E-mail: rector@ulstu.ru http://www.ulstu.ru
ОКПО 02069378, ОГРН 1027301160226
ИНН/КПП 7325000052/732501001

УТВЕРЖДАЮ
И.о. ректора УлГТУ
_____ А.Г. Пинков
26. 06. 2015 г.

АКТ

о внедрении в учебный процесс

результатов диссертационной работы Хородова В.С.

Результаты диссертационной работы Хородова В.С. «Разработка методов и средств многоагентного распределенного автоматизированного проектирования структурно-функциональных лингвистических моделей вычислительных устройств», представленной на соискание ученой степени кандидата технических наук, внедрены в учебный процесс Ульяновского государственного технического университета при обучении студентов направления «Информатика и вычислительная техника» по специальностям инженер 23010165, бакалавр техники и технологии 23010063, магистр техники и технологии 23010068 (лекции, лабораторные и практические занятия, курсовое и дипломное проектирование, выпускные работы на степень бакалавра и диссертационные работы на степень магистра).

Заместитель заведующего кафедрой «Вычислительная техника»
д.т.н., профессор

В.Н. Негода

ООО «Разработка кибернетических систем»,
432071 г. Ульяновск, Бородина 20, офис 24
тел. (8422) 72-48-88, 8 800-500-34-07

ОГРН 1067325004273
ИНН 7325059360 КПП 732501001
ОКПО 25510860

«Утверждаю»
Генеральный директор
ООО «Разработка
кибернетических систем»
Улыбин В.В.
2015 г.

АКТ

о внедрении результатов диссертационной работы

ХОРОДОВА ВИТАЛИЯ СЕРГЕЕВИЧА

Комиссия ООО «Разработка кибернетических систем» (г. Ульяновск) в составе:
Игонин Андрей Геннадьевич, технический директор,
Казаков Григорий Сергеевич, руководитель отдела разработки высоконагруженных систем,
Ключников Дмитрий Сергеевич, руководитель отдела перспективных разработок –
составила настоящий акт о том, что результаты диссертационной работы Хородова В.С. на тему «Разработка методов и средств многоагентного распределенного автоматизированного проектирования структурно-функциональных лингвистических моделей вычислительных устройств», представленной на соискание ученой степени кандидата наук, внедрены и использованы в ООО «Разработка кибернетических систем».

Разработанные в рамках диссертационной работы программные средства в виде агентов управления проектными задачами и агента базы данных использованы в процессе разработки, что обеспечило дополнительный контроль над ходом выполнения проектов. Применение методологии структурно-функциональных лингвистических моделей, заключающейся в наполнении библиотеки параметрическими функциональными модулями и их повторного использования позволило сократить время разработки на 7%.

Председатель комиссии,
технический директор, к.т.н. А.Г. Игонин

Члены комиссии:
руководитель отдела разработки высоконагруженных систем Г.С. Казаков
руководитель отдела перспективных разработок Д.С. Ключников