ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ПРИКЛАДНАЯ МАТЕМАТИКА

УДК 81'322

А.В. Пруцков, А.Н. Пылькин

ИНФОРМАЦИОННАЯ СИСТЕМА С ИСПОЛЬЗОВАНИЕМ ПОИСКА РЕШЕНИЙ ЗАДАЧ ГЕНЕРАЦИИ И ОПРЕДЕЛЕНИЯ В ПРОСТРАНСТВЕ СЛОВОФОРМ

Предложено представить решение задач генерации и определения в пространстве словоформ. Приведено соответствие терминологии теории графов и предложенного метода генерации и определения форм слов. Показано, что алгоритмы метода генерации и определения форм слов сводятся к поиску в глубину в графе в пространстве словоформ. Представление решения данных задач в пространстве словоформ позволило показать, что предложенные алгоритмы всегда находят решение задач генерации и определения форм слов.

***Ключевые слова**: автоматическая обработка текста, морфологический синтез и анализ, теория графов.*

Автоматическая обработка текстов и ее уровни. Задачи генерации и определения. Целью работы является разработка метода поиска решений задач генерации и определения в ориентированном графе в пространстве словоформ, основанного на предложенном представлении формообразования в виде цепочек преобразований.

Автоматическая обработка текстов представляет собой преобразование текста на естественном языке с помощью ЭВМ. Автоматическая обработка текстов используется при решении задач машинного перевода, проверки правописания, анализа текстов и выявления в них знаний, диалога ЭВМ с пользователем на естественном языке, анализа запросов в информационно-поисковых системах глобальных информационных сетей, анализа и синтеза речи.

Можно выделить три уровня обработки текстов:

1) семантический;
2) синтаксический;
3) морфологический.

На семантическом уровне связываются слова, сочетания и их смысловое значение. На синтаксическом уровне определяется взаимосвязь слов в предложении. На морфологическом уровне связываются слова предложения и их грамматические значения (падеж, время и т. п.).

Рассмотрим морфологический уровень обработки текстов. Морфологический анализ и синтез работают с предложениями, как с потоком (последовательностью) слов.

При морфологическом анализе для каждого слова решается задача определения. Определение заключается в нахождении по данной словоформе (форме слова) ее нормальной формы (основы) и грамматического значения.

Морфологический синтез включает решение для каждого слова задачи генерации. Генерация формы слова – это процесс получения формы слова с использованием в качестве начальных параметров основы и грамматического значения.

Автором статьи предложены метод и алгоритмы генерации и определения форм слов, позволяющие представить получение формы F из основы S и наоборот с помощью цепочек преобразований и не зависящие от вида и типа формообразования естественного языка [1]. Задачи генерации и определения форм слов являются задачами искусственного интеллекта (ИИ).

Решение задач искусственного интеллекта в пространстве состояний. Решение задач ИИ – это последовательность состояний, то есть ситуаций, складывающихся в процессе решения задачи. Оператор – некоторая последователь-

ность действий, преобразующая одно состояние в другое, то есть совершающая переход к следующему этапу решения задачи [2].

Решение задач ИИ можно представить в виде графа в пространстве состояний. Вершинами графа являются состояния, а дугами – операторы. Корневая вершина является начальной ситуацией, а конечные вершины – целевыми или тупиковыми. Целевые вершины приводят к решению задачи, а тупиковые вершины – нет. Тогда решение задачи можно рассматривать как последовательность операторов, которая преобразует начальную ситуацию в целевую вершину.

Для поиска в графе целевых вершин существует два простейших метода поиска в графах: поиск в глубину и поиск в ширину. Данные методы поиска предполагают последовательный перебор возможных альтернатив. Чтобы повысить эффективность поиска, вводится правило выбора очередной вершины или отсекаются заведомо тупиковые направления поиска, не приводящие к решению. Такой подход называется эвристическим поиском [3].

Решение задач генерации и определения в пространстве словоформ. Предлагается метод поиска решений задач генерации и определения в ориентированном графе в пространстве словоформ, основанный на предложенном методе генерации и определения. Пространство словоформ является модификацией пространства состояний и учитывает специфику задач генерации и определения.

Построение графа в пространстве словоформ основывается на следующем соответствии терминологии теории графов [4] и метода генерации и определения форм слов:

- вершины графа – это словоформы, при этом корневая вершина – основа, промежуточные вершины – промежуточные словоформы, а конечные вершины – словоформы парадигмы;

- дуги графа – это преобразования, которые необходимо произвести над одной формой, чтобы получить другую;

- путь (маршрут) в графе – это цепочка преобразований; путь от корневой вершины к конечной – прямая цепочка преобразований, путь от конечной вершины к корневой – обратная цепочка преобразований; можно выделить два типа путей:

1) целевой путь – целевая или ложная цепочки преобразований; целевая цепочка приводит к нахождению основы S, тип которой соответствует типу основы цепочки преобразований; ложная цепочка приводит к основе S, тип которой не соответствует типу основы цепочки преобразований;

2) тупиковый путь – тупиковая цепочка преобразований; тупиковая цепочка – это цепочка, которая прерывается из-за неприменимости преобразований к форме;

- пути, имеющие одну и ту же корневую вершину, составляют граф, и корневая вершина становится общей для них.

Оператор перехода из одного состояния в другое состоит из двух действий:

1) применения преобразования к текущей словоформе;

2) проверки условия перехода в следующее состояние, которое можно сформулировать двумя способами:

- относительно преобразования: если преобразование применимо к форме, то перейти в следующее состояние, иначе считать данный путь тупиковым;

- относительно формы: если преобразованная форма равна форме, соответствующей следующей вершине, то перейти в следующее состояние, иначе считать данный путь тупиковым.

Обе формулировки данных условий равнозначны.

За счет отсечения тупиковых путей сокращается количество рассматриваемых вершин, пройденных во время решения задач.

В графе в пространстве словоформ отсутствуют контуры и петли. Наличие контуров и петель в графе означало бы существование бесконечных словоформ, что невозможно в естественных языках. Даже если подстроки в словоформе повторяются, проще описать их последовательно, чем вводить контуры или петли и условия их окончания. Таким образом, граф в пространстве словоформ является бесконтурным.

В зависимости от задачи генерации и определения дуги в графе ориентированы в противоположные стороны. В случае задачи генерации дуги ориентированы от корневой вершины к конечным. В случае задачи определения дуги ориентированы в обратном направлении: от конечных вершин к корневой. Поэтому направления ориентации дуг на графах на рисунках не указываются.

При решении задачи генерации используются прямые преобразования (показаны на рисунках). В случае решения задачи определения используются задачи преобразования, обратные данным.

Граф в пространстве словоформ является ориентированным деревом со следующими свойствами:

- полустепень захода каждой вершины графа, кроме корневой, равна 1;

- полустепень исхода каждой вершины графа, кроме корневой и конечных, равна 1;

- полустепень исхода корневой вершины равна 0 для задачи определения и равна числу форм в парадигме для задачи генерации;

- полустепень исхода любой конечной вершины равна 1 для задачи определения и равна 0 для задачи генерации.

В пространстве словоформ находится не один граф, а несколько графов (рисунок 1), каждый из которых соответствует определенному типу основы T. Таким образом, слова имеют одинаковый тип формообразования, если их графы в пространстве словоформ будут идентичны.

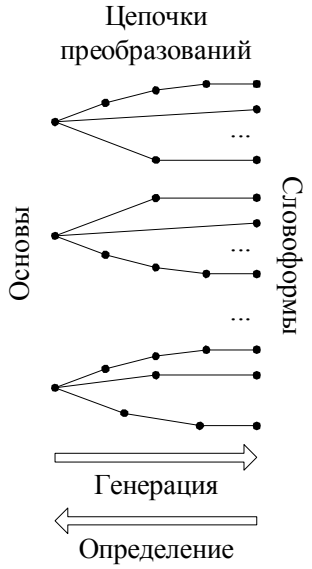


Рисунок 1 – Представление задач генерации и определения в виде графов в пространстве словоформ

Рассмотрим пример одного из таких графов в пространстве словоформ. На рисунке 2 представлен граф в пространстве словоформ для слова финского языка «*lintu*» – «*птица*». На рисунке жирным шрифтом выделены формы парадигмы с указанием падежа и числа: единственного (*yks.*) или множественного (*mon.*). Формы, подписанные шрифтом без выделения и обозначения падежа, являются промежуточными. В финском языке важное значение имеет основа родительного падежа (*gen.*) (в данном примере «*linnu*») [5], от которой образуются большинство остальных форм слова. Данное обстоятельство и позволило представить граф в виде дерева, в котором не только корневая вершина имеет несколько вершин.

Граф хранится как совокупность путей от корневой вершины к конечным вершинам по следующим причинам:

- большинство форм слов образуются путем добавления окончания, следовательно, только корневая вершина имеет более одной дочерней вершины;

- быстрый доступ к любому пути в графе.

Каждому пути в графе, как и цепочке преобразований, поставлены в соответствие тип основы T и грамматическое значение G.

В пространстве словоформ задача генерации представляет собой поиск пути от корневой вершины к конечной вершине, а задача определения заключается в нахождении такой конечной вершины, которая приведет к корневой вершине. При этом к форме слова применяются преобразования, приписанные дугам.

Задача генерации форм слов заключается в выборе графа, соответствующего типу основы T, и пути в данном графе, соответствующего грамматическому значению G. Таким образом, путь к решению задачи в графе известен, и поэтому выполняются только те шаги (преобразования), которые ведут к решению. Целевая вершина, представляющая собой искомую словоформу, находится за минимальное количество шагов. Поэтому поиск решения задачи генерации является эвристическим.

Задача определения сводится к перебору графов, соответствующих различным типам основ T, и поиску в них путей от конечных вершин к корневым. Если достигнута корневая вершина и соответствующая ей основа S присутствует в списке основ, то можно сделать вывод, что данная основа S и грамматическое значение G, соответствующее пути в графе, являются решением задачи определения. Поиск в графе в пространстве словоформ решения задачи определения является эвристическим. При этом каждая вершина рассматривается не более одного раза по следующим причинам:

- если преобразование неприменимо к форме, то переход к следующим вершинам данного пути не происходит;

- откат к родительской вершине не происходит, так как в случае тупикового пути необходимо прекратить рассмотрение текущего пути и перейти к следующей конечной вершине, а следовательно, к следующему пути.

Оценим количество графов и путей в пространстве словоформ для слов русского языка. В работе [6] все слова русского языка разделены на 116 флективных классов, что равно количеству графов. Суммарный объем парадигм этих классов составляет порядка 3800 форм, что равно суммарному количеству путей в этих графах (цепочек преобразований).

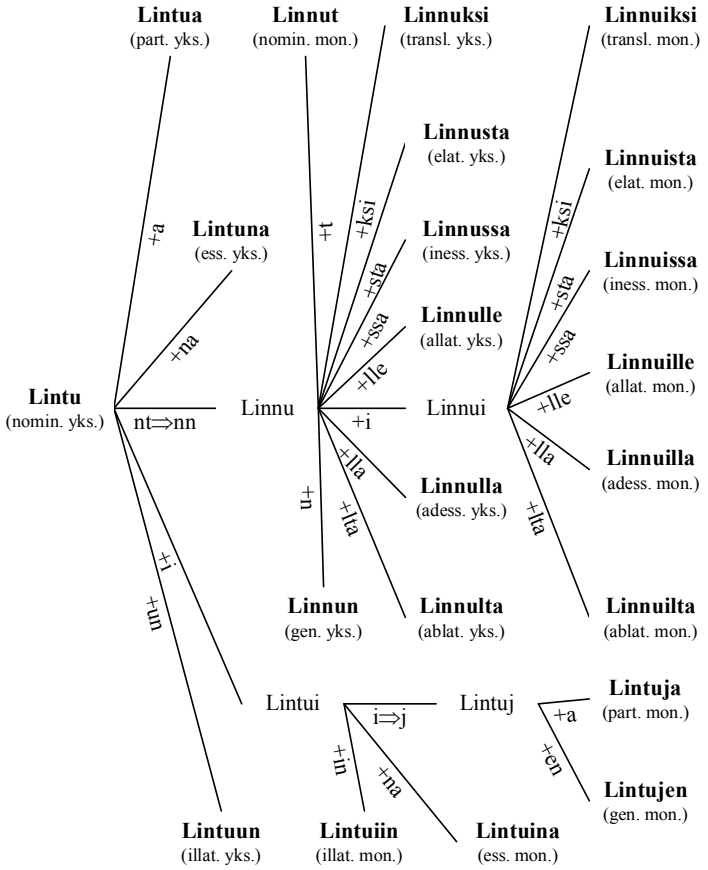


Рисунок 2 – Пример графа в пространстве словоформ для слова финского языка «*lintu*» – «*птица*»

Данные количественные характеристики объясняют более низкую скорость определения словоформ программной системы, основанной на предложенном методе, по сравнению с аналогичными системами (см. таблицу). Низкая скорость работы является платой за универсальность данного метода, его независимость от формообразования естественных языков.

Таблица

Сравниваемые системы	Тип и частота процессора	Скорость определения, тыс. слов/с
МЛ Морфология SDK (компания Медиалингва)	Intel Pentium, 0,2 ГГц	0,5
RCO Morphology Professional SDK (компания RCO)	AMD Athlon, 1 ГГц	100
Система, основанная на предлагаемом методе	Intel Pentium IV M, 1,8 ГГц	0,7

Заключение. Предложенный поиск решений задач генерации и определения представляет собой модифицированный в соответствии со структурой графа и особенностями задачи поиск в глубину. Метод поиска в глубину предпочтительней метода поиска в ширину. При поиске решения задачи генерации путь известен заранее, поэтому нет необходимости рассматривать другие альтернативы. При поиске решения задачи определения приходится обходить все пути, поэтому нет разницы, какой метод использовать – поиск в глубину или поиск в ширину, однако поиск в ширину требует много памяти для хранения списка пройденных вершин.

Метод поиска в глубину всегда находит решения задачи или доказывает, что решение не существует. Поэтому алгоритмы генерации и определения форм слов, основанные на методе поиска в глубину, всегда найдут решение задачи, если оно существует, что доказывает правильность алгоритмов генерации и определения форм слов.

Проведена оценка количества графов и путей в пространстве словоформ для слов русского языка.

Метод поиска решения задач генерации и определения в пространстве словоформ позволяет просто, понятно и наглядно представить предложенный метод и алгоритмы генерации и определения форм слов.

Библиографический список

1. *Пруцков А.В.* Генерация и определение форм

слов естественных языков на основе их последовательных преобразований // Вестник Рязанского государственного радиотехнического университета. Вып. 27. / Рязан. гос. радиотехн. ун-т. – Рязань, 2009. – С. 51-58.

2. *Нильсон Н.* Искусственный интеллект: методы поиска решений / пер. с англ. В.Л. Стефанюка; под ред. С.В. Фомина. – М.: Мир, 1973. – 271 с.

3. *Лорьер Ж.-Л.* Системы искусственного интеллекта: пер. с франц. – М.: Мир, 1991. – 568 с.

4. Дискретная математика. Теория графов: учеб. пособие / А.М. Гостин, В.П. Корячко; Рязан. гос. радиотехн. ун-т. – Рязань, 2006. – 80 с.

5. *Чернявская В.В.* Учебник финского языка. – СПб.: Виктория плюс. Аспет Плюс, 2002. – 320 с.

6. *Белоногов Г.Г., Богатырев В.И.* Автоматизированные информационные системы / под ред. К.В. Тараканова. – М.: Сов. радио, 1973. – 328 с.

УДК 621.317.75:519.2

В.В. Белов, М. В. Наумович

ПРОБЛЕМЫ РЕАЛИЗАЦИИ ФУНКЦИОНАЛЬНОСТИ СОБЫТИЙ В ПРОГРАММНЫХ СРЕДСТВАХ ИНДИВИДУАЛЬНОГО ПЛАНИРОВАНИЯ

Приводится обзор современных средств планирования действий. Дана классификация и указаны пути развития функциональности индивидуальных планировщиков.

Ключевые слова: *органайзер, событие, планирование, проблема, представление.*

Введение. В современном быстро изменяющемся мире люди вынуждены постоянно воспринимать большой объем разнообразной информации, на ее основе вырабатывать решения и планировать свои действия. Например, руководящим работникам от менеджеров низших звеньев до высших управленцев государственного аппарата в течение трудового дня приходится решать множество задач, проводить значительное количество встреч, совещаний, присутствовать на различных мероприятиях. Предварительное планирование дел помогает повысить эффективность любой деятельности как личной, так и профессиональной. Начальной основой для этого служили ежедневник, а также список АВС приоритетов и список задач To-Do. Впоследствии все эти компоненты были соединены в органайзере - небольшой книге, содержащей календарь, адресную книгу и блокнот. Со временем специальные программные средства пришли на смену традиционному бумажному органайзеру [1].

Цель работы – обзор существующих программных средств планирования деятельности и определение возможных направлений и механизмов их развития.

Обзор современных органайзеров. Современные органайзеры можно классифицировать следующим образом.

❖ Карманные: приложения для КПК и смартфонов.

❖ Настольные: программные средства для десктопов и ноутбуков.

❖ Онлайн: автономный веб-инструментарий для взаимодействия через интернет-браузер.

Настольные органайзеры обладают наиболее широкой функциональностью.

Thunderbird (3.1) с расширением Lightning (1.0) от компании Mozilla – это популярный свободный кроссплатформенный почтовик с плагином-органайзером на основе открытого исходного кода [7]. Программа позволяет вести учет событий и задач. Форма добавления новой задачи заключает в себе наиболее широкие функциональные возможности (рисунок 1).

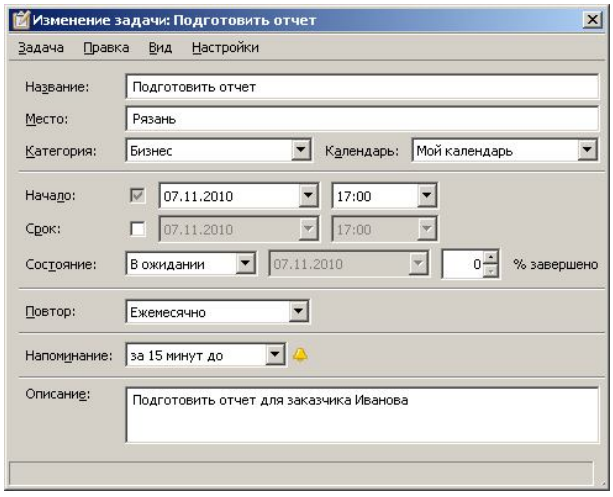
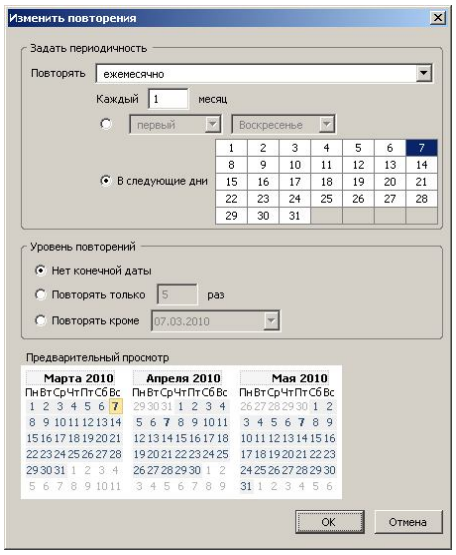


Рисунок 1 – Форма добавления новой задачи в Thunderbird

Основными атрибутами являются название, место и описание создаваемой задачи. Ей также можно назначить календарь, категорию, приоритет и уровень приватности (если речь идет о совместном использовании календаря). Присутствует возможность прикрепить вложения. Относительно временных рамок событие может иметь начало, начало и окончание исполнения, либо не иметь ни того, ни другого. В последнем случае программа не позволит установить напоминание [8]. Для задания режима повторения помимо списка готовых вариантов предусмотрена специальная форма (рисунок 2,а). Она позволяет довольно гибко настроить график повторений. Для установки напоминаний также имеется отдельная форма (рисунок 2,б). Основным недостаткам является плохо реализованная работа с категориями, или тегами. Событию можно назначить только одну категорию.



а

б

Рисунок 2 – Настройки задачи в Thunderbird:
а – форма задания повторения;
б – форма задания напоминания

Outlook (2010) от компании Microsoft – один из наиболее распространенных органайзеров. Это платное программное обеспечение для платформы Microsoft Windows существенно оптимизировано для совместной работы группы людей, однако его можно использовать и в персональных целях.

Outlook позволяет создавать два типа событий: задача и собрание или встреча [2]. Форма ввода задачи представлена на рисунке 3.

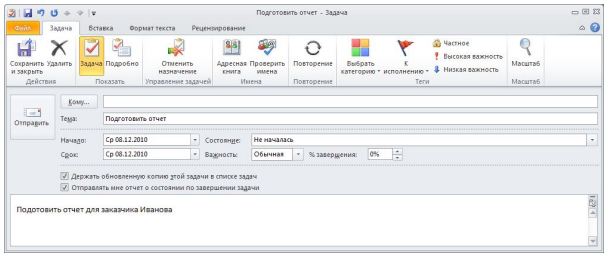


Рисунок 3 – Форма ввода задачи в Outlook

Функциональность событий несколько отличается. У задачи нет параметра места и во времени ее можно позиционировать лишь с точностью до дня [9]. Также можно описать бессрочные задачи, однако если определить дату начала, то необходимо указать и дату завершения. Собрание всегда требует определенности его во времени, не имеет приоритетности и не позволяет отслеживать состояние [3]. Существенное отличие от Thunderbird заключается в упрощенном выборе единичных напоминаний из списка предустановок. Главным недостатком реализованной функциональности является четкое размежевание типов событий и невозможности их интеграции, а также примитивная система напоминаний.

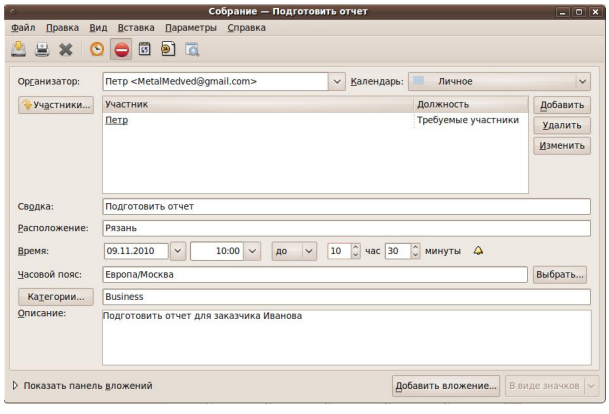


Рисунок 4 – Формы ввода собрания в Evolution

Evolution (2.32) – распространенный органайзер-почтовик для ОС Linux благодаря тому, что входит в состав графической оболочки GNOME [10]. Для представления событий предлагает использовать встречи (собрания) и задачи. Помимо набора базовых свойств доступны гибкие настройки повторений (подобно

Thunderbird) и напоминаний (оповещение с помощью окна, звука, запуска программы, отправки почты). Встреча всегда требует точного описания временных рамок, для задачи, напротив, можно их частично или полностью не указывать. Серьезным недочетом функциональности является отсутствие механизма оповещений для задач.

Contactizer Pro (3.8) – органайзер для Mac OS по сравнению с предыдущими предоставляет более мощные возможности [11]. Помимо традиционных встреч и задач присутствует возможность создания проектов для трехуровневой организации представления событий. Представление данного вида события показано на рисунке 5. Проект помимо собственных стандартных атрибутов содержит еще свои шаги или подпункты, каждый из которых, в свою очередь, может содержать наборы задач, встреч и прочих данных. Основные недостатки реализации события проект заключаются в представлении его подуровней. Нельзя создавать вложенные проекты и представлять подпункты в произвольном порядке. Шаги также обладают всего несколькими атрибутами: название, дата выполнения, статус и заметка. Таким образом, введение данного подтипа выглядит малопродуктивным.

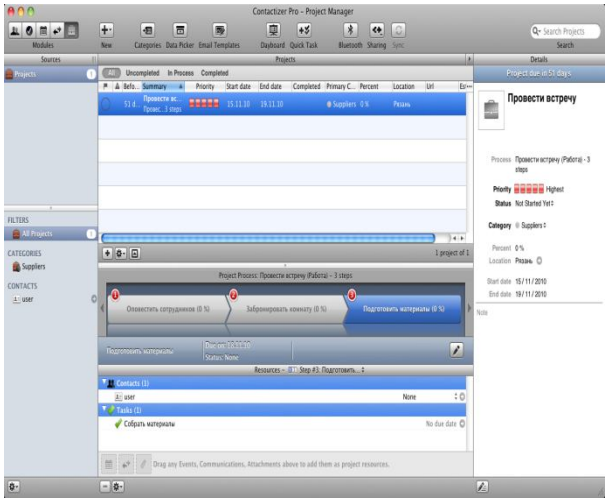
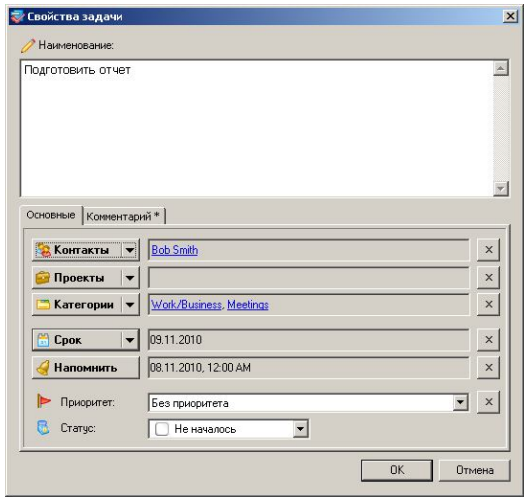


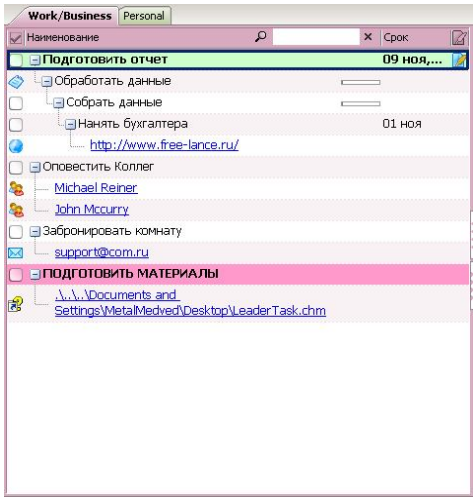
Рисунок 5 – Представление проекта в Contactizer Pro

Представление встреч и задач аналогично Thunderbird и Evolution. Помимо прочего стоит отметить улучшенную пятиуровневую систему приоритетов, гибкие настройки напоминаний и возможность организации ссылок между событиями.

LeaderTask (6.9.7) – органайзер от российских разработчиков представляет совершенно новые способы планирования [12]. В нем всего один основной вид представления событий – задача. Она обладает базовым набором характеристик, за исключением местоположения, определяется во времени лишь моментом выполнения (рисунок 6,а). Основным преимуществом данного продукта является возможность вкладывать задачи одна в другую, создавая многоуровневые события высокой степени вложенности, однако программа допускает лишь древовидные связи; дополнительно для большей наглядности в представление можно включать заметки и ссылки (рисунок 6,б). В программе также есть понятие проект, однако оно несет в себе лишь некоторые формальности для организации методологий тайм-менеджмента [4], так как фактически равноценно вложенным задачам. Для каждой задачи ведется история, в которую можно вносить записи, связанные с процессом выполнения. Хорошо проработана система тегов: пользователь может самостоятельно редактировать папки-категории, приоритеты и собирать задачи в проекты. К недостаткам относятся простая система напоминаний, а также недостаточная информационная нагрузка модели проектов.



а



б

Рисунок 6 – Органайзер LeaderTask: а – форма ввода задачи; б – древовидное представление задач

Рассмотрим онлайн-органайзеры.

Календарь от Google является одним из самых популярных на сегодняшний день web-сервисов. Большинство настольных и карманных органайзеров поддерживают синхронизацию с этим сервисом. Он позволяет создавать мероприятия и задачи [5, 6]. Набор параметров у обоих весьма ограничен (рисунок 7). Событие мероприятие требует четкого определения во времени, а у задачи есть всего два поля: дата выполнения и заметка. К особенностям данного продукта относятся: 1) довольно развитый механизм оповещений и повторений; 2) возможность приглашения участников встречи посредством приглашений, рассылаемых по электронной почте; 3) полное отсутствие тегов и приоритетов; 4) интеграция с сервисом карт Google (возможны отметки места события).

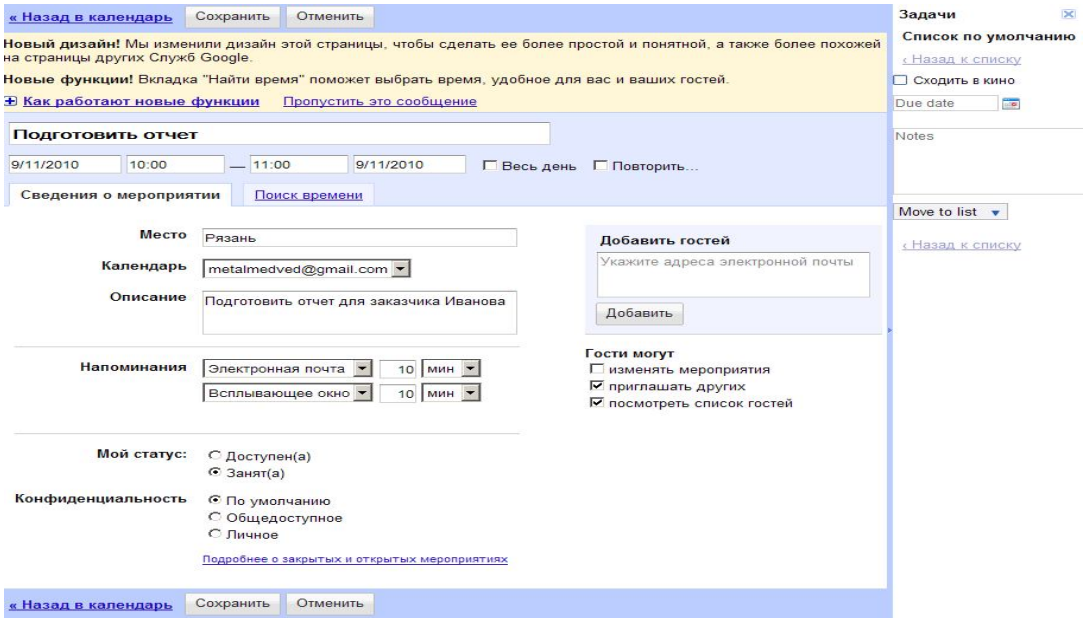


Рисунок 7 – Фрагмент формы создания мероприятия и лист задач в Календарь Google

Календарь способен отправлять напоминания о событиях по e-mail и через SMS. Работа осуществляется в окне браузера через вебинтерфейс, данные хранятся на централизованном сервере Google, поэтому получить доступ к расписанию можно с любого компьютера, подключенного к Интернет (при этом данные защищены паролем) [5].

В интерфейсе можно пользоваться «горячими клавишами». Весьма полезным является механизм строки для быстрого формирования события. Например, если разместить в этой строке «Today Call Mr. Smith at 3pm», то автоматически будет создано событие на текущий день в 15:00 с названием «Call Mr. Smith» («Позвонить мистеру Смиту»). Строки быстрого формирования событий могут не только вводиться вручную, но и извлекаться из писем.

Одновременно можно создавать несколько календарей и автоматически помечать официальные праздники [13].

Важнейшим достоинством Календаря Google является возможность совместного использования календаря несколькими пользователями. Кроме того, календарь можно сделать доступным только для избранных пользователей и осуществлять планирование общих встреч. Календарь реализован с использованием технологий JavaScript и AJAX, резко сокративших время отклика системы и обеспечивших достижение эффекта программы, установленной локально.

На текущий момент интерфейс предоставляется на семнадцати языках, включая русский, почти все европейские и даже японский и китайский. Сведения в календарь можно вводить практически на любом языке.

Онлайн органайзер Toodledo пока не получил широкого распространения в России и не русифицирован. Однако объединяет в себе многие достоинства не только похожих ресурсов, но и настольных приложений [14]. Регистрация бесплатна, однако существуют платные услуги.

Форма ввода заданий обладает множеством параметров (рисунок 8). Помимо базового набора существуют некоторые особенные свойства. Поле «Context» задает характер задания, «Tag» содержит ключевые слова. Отдельно выделены 3 поля «Start Date», «Due Date» и «Length». Каждое из них может содержать значение или быть пустым.

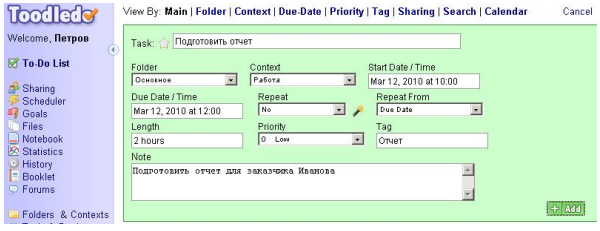


Рисунок 8 – Фрагмент формы ввода задания в органайзере Toodledo

Система повторов и напоминаний организована оригинальным образом (рисунок 9).

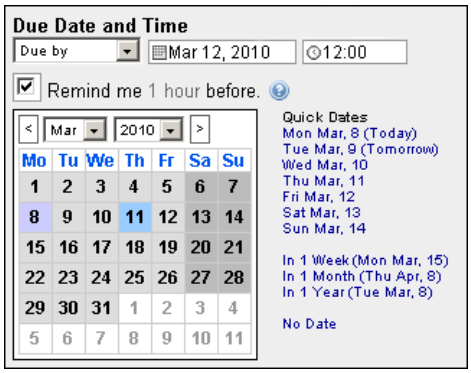


Рисунок 9 – Задание даты выполнения и напоминания в Toodledo

Она построена целиком вокруг поля «Due Date», то есть срока выполнения задачи. Это может оказаться весьма неудобным при традиционных способах использования органайзеров.

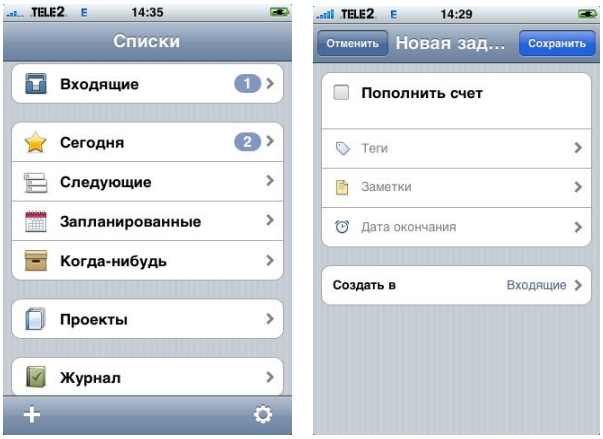
Планировщик Toodledo обладает несколькими уникальными в своем роде возможностями. Во-первых, в нем реализована возможность ведения статистики выполнения задач. Во вторых, в платную подписку включен «Scheduler». Это планировщик, который может помочь скомпоновать список дел на определенный промежуток свободного времени.

Рассмотрим категорию карманных планировщиков. Наиболее функциональные приложения этой области, как правило, имеют средства синхронизации со своими же настольными вариантами и (или) онлайн органайзерами.

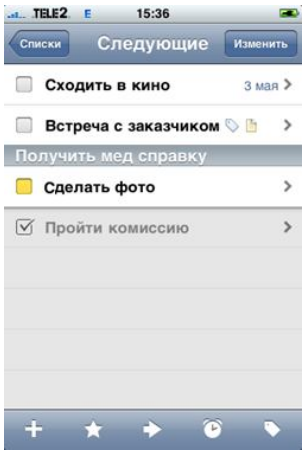
Things (1.4) – планировщик для iOS, имеется также настольная версия для Mac OS [15]. Это платное приложение, есть локализация для России. Все функции программы скомпонованы в главном меню, которое состоит из семи пунктов: входящие, сегодня, следующие, запланированные, когда-нибудь, проекты и журнал (рисунок 10,а). Рассмотрим их подробно.

Первые пять категорий – номинальные. Фактически задачи в них представляются одними и теми же средствами. Каждое событие может содержать заголовок, теги – ключевые слова, заметки для более подробного описания и дату, к который нужно успеть выполнить задачу (рисунок 10,б). Существенными недостатками являются отсутствие возможности создания повторяющихся событий и привязки мультимедийных данных.

Внутри каждой категории задачи представляются в виде списков (рисунок 10,в). С помощью нижнего меню можно добавлять новое событие кнопкой «+». Отметив задачу звездочкой, мы перемещаем его в пункт «сегодня». Так же можно переносить задачи между пунктами и фильтровать по дате и тегам. Законченные события помечаются галочкой.



а б



в

Рисунок 10 – Интерфейс Things: а – главное меню; б – добавление задачи; в – список задач

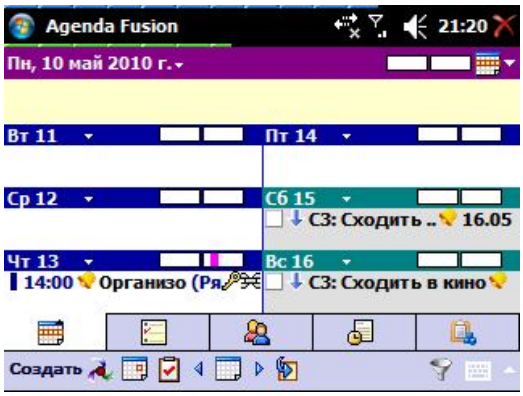
Пункт «проекты» предназначен для хранения сложных дел, состоящих из нескольких подзадач. Каждый проект может содержать только один подуровень. Все подзадачи проектов отображаются в пункте «следующие», подразделяясь на группы (рисунок 10,в).

Пункт «журнал» используется для логирования выполненных событий. По желанию задачи в него могут переноситься раз в день или после перезапуска программы.

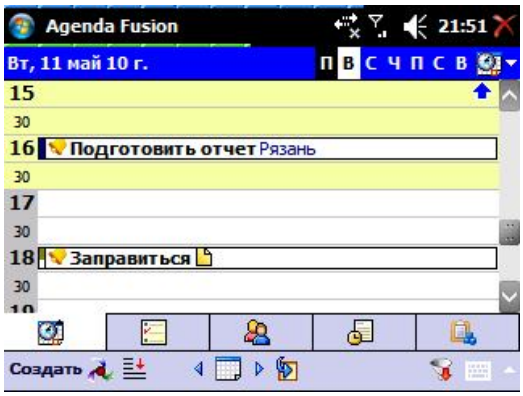
Agenda Fusion (8.7) – один из наиболее известных органайзеров для платформы Windows Mobile [16]. Этот коммерческий продукт официально не имеет русского языкового интерфейса. Программа синхронизируется с почтовой службой и адресной книгой телефона, что позволяет использовать в планировании дополнительную информацию.

Главное окно программы состоит из пяти вкладок и меню (рисунок 11,а). Основная вкладка №1 содержит общее расписание запланированных событий. Можно выбрать множество различных способов отображения: планы на

день (рисунок 11,б), неделю (рисунок 11,а) или месяц. Программа позволяет создавать события четырех типов, для каждого из которых отведена отдельная вкладка, так как на все виды отображаются на основной вкладке.



а



б

Рисунок 11 – Основная вкладка Agenda Fusion. Различные способы представления интерфейса

Наиболее простой тип – напоминание (alarm). Эта группа событий отображается на вкладке №4. Она ориентирована на простые разовые задачи. При вводе необходимо задать заголовок и время, через которое следует напомнить (рисунок 12). Дополнительно можно ввести текстовую, графическую или аудиоинформацию.

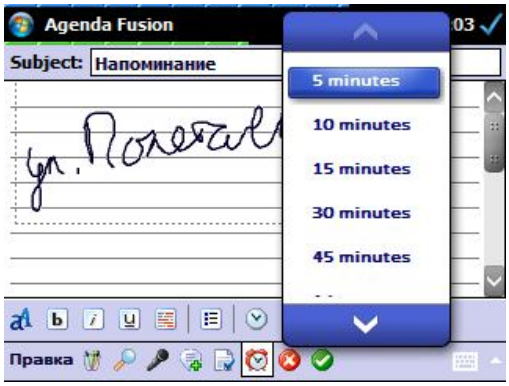


Рисунок 12 – Событие «Напоминание»

Следующие два вида событий: задачи (tasks) и встречи (appointments) очень похожи между собой. Различие заключается в том, что в задачах нельзя указывать временной интервал события. Рассмотрим свойства встречи. Прежде всего, это основные параметры: название, временной интервал, место и категория (рисунок 13). Функциональность программы позволяет настроить весьма сложные схемы. Можно ввести заметки и присвоить встрече категории. Имеется возможность связки с событием различных объектов: других событий, контактов из адресной книги, файлов или веб-ссылок.

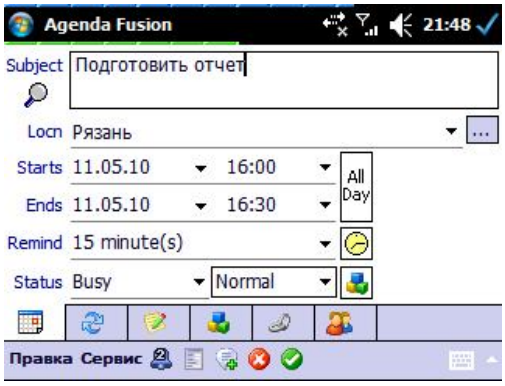
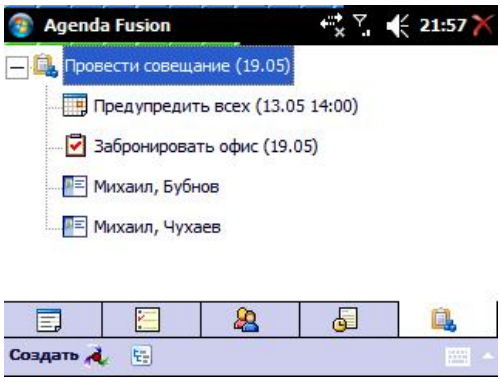
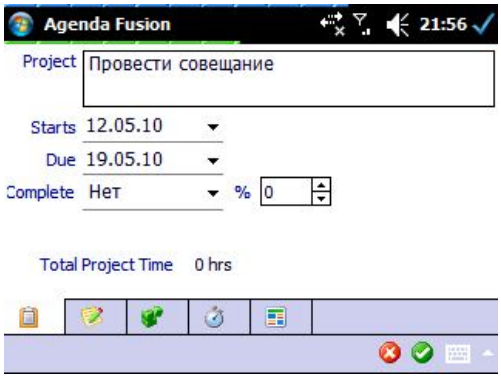


Рисунок 13 – Событие «Встреча»

Последним типом представляемых событий являются проекты (projects) (рисунок 14,а). Фактически проект служит контейнером для связанных событий вышеописанных типов, а также некоторых дополнительных объектов. При его создании задаются основные параметры проекта (рисунок 14,б), присутствует возможность ведения и просмотра времени, потраченного на проект.



а

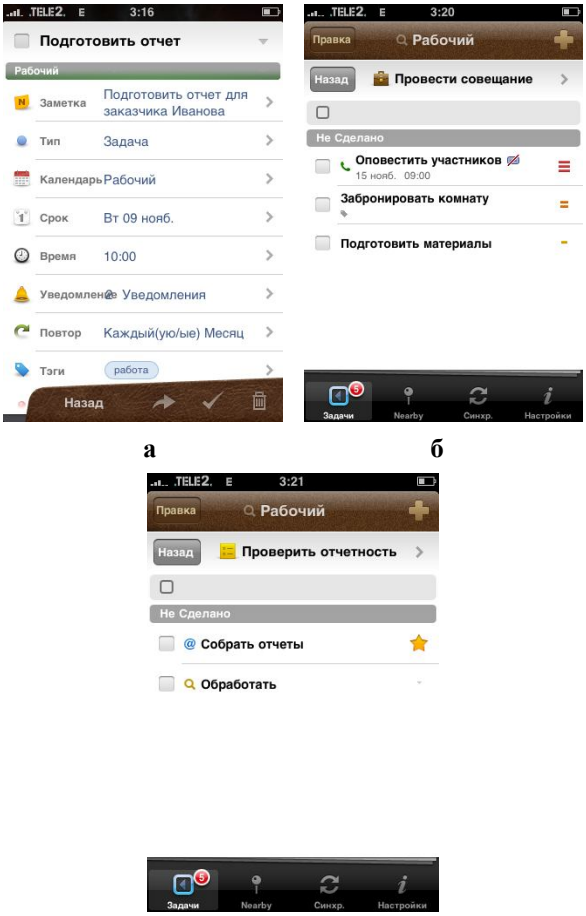


б

Рисунок 14 – Событие проект. Основные вкладки

В заключение следует отметить возможность фильтрации отображаемых событий по присвоенным им категориям, огромное количество настроек программы, с помощью которых можно изменять почти каждый элемент интерфейса, а также возможность отправки событий по внешним интерфейсам на другие устройства и синхронизацию с Outlook mobile.

2Do – органайзер для iOS. Позволяет создавать задачи, проекты и списки (рисунок 15). Все они обладают одинаково широким набором свойств и определяются во времени сроком выполнения. Проекты и списки фактически применяются для одной и той же цели – двухуровневой организации задач – и отличаются лишь смысловой нагрузкой. Особо стоит отметить работу данного приложения с параметром местоположения задачи. В программе есть функция ручного поиска задач, местоположение которых находится в заданном радиусе от пользователя, однако никаких дополнительных фильтров (хотя бы по временным параметрам) при этом задействовать нельзя. Имеются широкие возможности синхронизации с Outlook для Windows, iCal для Mac и Toodledo.



а **б**

в

Рисунок 15 – Представление событий в 2Do:
а – задача; б – проект; в – список

2Do имеет ряд замечательных особенностей, в числе которых: возможность прикрепления голосовых заметок к задачам; управление по меткам из слов или контактов; легкое перемещение задач, проектов или контрольных списков из одного календаря в другой [17]. Задачи, перемещенные в календарь сегодня, автоматически получают дату обновления до текущего дня.

Заключение. На основе проведенного анализа программных средств индивидуального планирования можно выделить ряд базовых проблем представления событий.

❖ Программы-планировщики и органайзеры в основном работают с событиями, у которых задан момент начала действия. В таблице приведена классификация событий относительно их основных временных характеристик.

Классификация событий
по временным параметрам

Вид	Начало	Конец
I	+	-
II	+	+
III	-	-

Современные средства в основном способны работать с событиями вида I. Причем в некоторых программах для обозначения начала события используется термин «срок» (LeaderTask, 2Do). Поддержка данного вида события сегодня в совершенстве реализована в любом планировщике.

Событие вида II косвенно определяет длительность Δt_i временного интервала, в течение которого происходит событие. При этом предполагается, что $\Delta t_i \le t_{fin} - t_{start}$, где t_{start} – момент начала события; t_{fin} – момент, не позднее которого событие должно быть завершено. Данный вид довольно широко представлен в современных планировщиках (Thunderbird, Outlook, Календарь Google, Agenda Fusion), однако из-за отсутствия гибкой системы напоминаний только Thunderbird позволяет эффективно им пользоваться.

Событие вида III не имеет определенных временных рамок. Практически такие события представимы во всех современных органайзерах, однако ни одна программа не в состоянии напоминать о них, поскольку нет основы для формирования напоминаний – момента начала события. Определяющим фактором таких событий в отсутствии временных рамок становится *контекст* текущей ситуации – место и/или тип действия (звонок, встреча) в совокупности с приоритетом относительно других событий. *Опираясь на эти данные, можно автоматически назначить событиям вида III временные рамки при составлении плана действий для пользователя.*

Заметим, что возможны ситуации, когда событие обладает такими свойствами, что для его планирования необходимо указать только момент завершения. Такие события легко сводятся к событиям вида II путём указания в качестве начала текущего момента времени.

❖ Параметр местоположения очень часто играет значительную роль в описании задачи. Однако лишь немногие современные программы в какой-то мере поддерживают его обработку усилиями пользователя. *Этот процесс требует автоматизации*, которая позволит реализовать совершенно новую систему напоминаний. На текущий день созданы все предпосылки для реализации оперативного выявления положения пользователя в пространстве. Уже сегодня все необходимые данные для расчетов можно получать посредством широко распространенных модулей GPS, ГЛОНАСС и локальных средств позиционирования на основе ультразвука и инфракрасного излучения.

❖ Свойство повторения напоминаний везде реализовано на основе циклов с равными промежутками времени. Практические задачи зачастую требуют более сложную модель. Например, *система логического вывода на основе онтологий, использующая в качестве знаний дополнительные слоты событий (местоположение и категории действий), позволит реализовать эффективные механизм выдачи напоминаний для событий вида II и III.*

❖ Связка событий в большинстве современных планировщиков реализована хаотично (Contactizer Pro, Agenda Fusion) или упрощенно (LeaderTask). Между тем этот механизм требует чётко организованной структуры и удобного представления в интерфейсе пользователя. Например, это может быть граф – сеть с отношениями приоритетности. *Упорядоченные связки событий позволят осуществлять автоматизированное составление оптимального плана действий для пользователя.*

Классы традиционно реализуемых в современных планировщиках событий – встреч и задач, а также их производных – обладают каждый отдельной функциональностью. *Для повышения гибкости представления необходимо объединить возможности всех классов в одном.* Это также позволит унифицировать и упростить процесс планирования.

Библиографический список

1. Мобильные планировщики // Chip 2009. - №11. – С. 122-125.

2. *Преппернау Дж., Кокс Дж.* Microsoft Office Outlook 2007. Шаг за шагом. М.: Эком, 2007. 544 с.

3. *Спиридонов О.* Outlook 2007: возможности для офиса // КомпьютерПресс. 2007 № 2. – С. 23-30.

4. *Allen D.* Getting Things Done: The Art of Stress-Free Productivity // Penguin Group (USA) Incorporated. 2001 – 267 pages.

5. *Письменный А.* Календарь от Google // Компьютера. 2006 - №223. – С. 56.

6. *Хализев В.* Обзор программ-органайзеров // Hard'n'Soft. 2008 - №5. – С. 34-36.

7. iXBT: Обзор Mozilla Sunbird и Chatzilla [Электронный ресурс]. URL:http://www.ixbt.com/soft/mozilla-sunbird-chatzilla.shtml (дата обращения 04.04.2010).

8. Calendar - Frequently Asked Questions [Электронный ресурс]. URL: http://www.mozilla.org/projects/calendar/faq.html (дата обращения 01.10.2010).

9. *Крупин А.* Office 2010 Beta: обзор новых возможностей [Электронный ресурс]. URL:http://www.computerra.ru/terralab/softerra/479603 (дата обращения 28.09.2010).

10. Evolution 2.30 User Guide [Электронный ресурс]. URL: http://library.gnome.org/users/evolution/stable (дата обращения 05.10.2010).

11. Introducing Contactizer Pro 3.8 [Электронный ресурс]. URL: http://www.objective-decision.com/en/products/contactizerpro (дата обращения 03.10.2010).

12. Обзор Возможностей планировщика LeaderTask. [Электронный ресурс]. URL: http://www.leadertask.ru/content/view/21/36 (дата обращения 30.09.2010).

13. Google календарь – статьи справки [Электронный ресурс]. URL: http://www.google.com/support/calendar/?hl=ru (дата обращения 04.04.2010).

14. Toodledo help [Электронный ресурс]. URL:http://www.toodledo.com/info/help.php (дата обращения 05.04.2010).

15. *Егорова К.* Things 1.2: как стать организованнее? [Электронный ресурс]. URL:http://www.macster.ru/review/things-12-kak-stati-organizovannee (дата обращения 03.10.2010).

16. Agenda Fusion 8 for Pocket PC. Manage your time like a Pro. [Электронный ресурс]. URL: http://www.developerone.com/agendafusion (дата обращения 10.10.2010).

17. *Фаттахов Э.* [Обзор] 2Do. [Электронный ресурс]. URL: http://icoderu.wordpress.com/2010/02/05/89 (дата обращения 12.10.2010)

УДК 519.716

В.В. Тарасов, В.А. Саблина

О ПОРЯДКАХ МАКСИМАЛЬНЫХ ЗАМКНУТЫХ КЛАССОВ В АЛГЕБРЕ ЧАСТИЧНЫХ БУЛЕВЫХ ФУНКЦИЙ

Исследованы порядки всех максимальных подалгебр (предполных классов) алгебры частичных булевых функций; для шести классов найдены порядки и получены конечные базисы; два класса не имеют конечного базиса.

Ключевые слова: *частичные булевы функции, не всюду определенные функции алгебры логики, выразимость в алгебре логики.*

Введение. Проблема функциональной выразимости в алгебре логики была решена работой Э. Поста в 1921 г. и окончательно в виде монографии в 1941 г. [1, 2]. В 1966 г. вышла книга [3] «Функции алгебры логики и классы Поста», которая вот уже почти полвека является настольной книгой советских и российских математиков-дискретчиков и инженеров-кибернетиков. Описание, аналогичное постовской решетке замкнутых классов, в трехзначной логике оказалось весьма трудным из-за того, что, как оказалось, мощность замкнутых систем там равна континууму. Далее последовало фрагментарное исследование решетки многозначных логик; отметим лишь работы [4-11], которые в той или иной мере методически относятся к настоящей работе.

Обратим внимание на существование промежуточных итерационных систем, находящихся между алгеброй логики P_2 и трехзначной логикой P_3. Это, например, алгебра частичных булевых функций, введенная А.И. Мальцевым [10] как *специальная* подалгебра итеративной алгебры Поста трехзначной логики. В 1966 г. Р.В.Фрейвалд [11] решил проблему функциональной полноты в алгебре $\widetilde{P}_2$ – частичных булевых функций, найдя все ее предполные классы. Следуя логике развития теории P_3:

1) доказательство теоремы о полноте в P_3 [4];

2) определение порядков предполных классов в P_3 [5];

3) определение всех субмаксимальных классов в P_3 [6];

4) доказательство конечности порядков субмаксимальных классов P_3 [7-9],

мы выполним пункт 2 для алгебры $\widetilde{P}_2$.

Кроме теоретического обоснования решения поставленной в этой статье задачи, укажем некоторую практическую целесообразность. Приведем пример. Внешняя дизъюнкция СДНФ булевой функции работает лишь на наборах

$$(0,\ldots,0),(1,0,\ldots,0),(0,1,0,\ldots,0),\ldots,(0,\ldots,0,1), \quad (1)$$

на остальных же наборах дизъюнкция вполне могла бы быть реализована ущербно, как угодно, лишь бы оказаться дешевле и проще. Иначе дизъюнкция может быть заменена частичной функцией, совпадающей с дизъюнкцией в области определенности (1).

Цель работы. В настоящей работе найдены порядки для шести максимальных классов (с конечными базисами) алгебры $\widetilde{P}_2$ частичных булевых функций и показано, что остальные два класса не имеют конечного базиса. Порядком замкнутого класса с конечным базисом называется минимум из порядков всевозможных базисов этого класса [3]. Решение поставленной задачи дает надежду на некоторый толчок в развитии теории структуры замкнутых систем частичных функций. Также авторы хотят обратить внимание инженеров-схемотехников на некоторую возможную пользу от изготовления дешевых частичных функциональных элементов с целью использования их в синтезе схем.

Порядок класса $\widetilde{T}_0$. Класс всех частичных булевых функций, сохраняющих предикат $x \neq 1\,(x \in \{0,1,2\})$, обозначается как $\widetilde{T}_0$, поскольку он является расширением класса всех булевых функций, сохраняющих 0 (обозначения С.В. Яблонского [4]). Неопределенные места значений функции заполняются символами 2. Суперпозиция частичных функций на наборе примет значение 2 всякий раз, когда на этом наборе принимает значение 2 любая функция суперпозиции.

Везде в дальнейшем $\tilde{x} = (x_1,\ldots,x_n)$, n – количество аргументов функции. Рассматриваем систему функций из $\widetilde{T}_0$:

$$x \oplus y,\ xy,\ x \vee y = x \oplus y \oplus xy,$$

$$\varphi_{\vee}(x,y)=\begin{cases}2 \text{ при } (x,y)=(0,0),\\ 0 \text{ в других случаях;}\end{cases}$$

$$\varphi_{\&}(x,y)=\begin{cases}2 \text{ при } (x,y)=(1,1),\\ 0 \text{ в других случаях;}\end{cases}$$

$$\varphi_{01}(x,y)=\begin{cases}2 \text{ при } (x,y)=(0,1),\\ 0 \text{ в других случаях.}\end{cases}$$

$$\varphi^n_{\vee}(x_1,...,x_n)=\varphi_{\vee}(x_1\vee...\vee x_{n-1},x_n)=$$
$$=\begin{cases}2 \text{ при } \tilde{x}=\tilde{0},\\ 0 \text{ в других случаях;}\end{cases}$$

$$\varphi^n_{\&}(x_1,...,x_n)=\varphi_{\&}(x_1\,\&...\&\,x_{n-1},x_n)=$$
$$=\begin{cases}2 \text{ при } \tilde{x}=\tilde{1},\\ 0 \text{ в других случаях;}\end{cases}$$

$$\varphi^n_k(x_1,...,x_n)=$$
$$=\varphi_{01}(x_1\vee...\vee x_k,x_{k+1}\,\&...\&\,x_n)=$$
$$=\begin{cases}2 \text{ при } \tilde{x}=(\underbrace{0...0}_{k \text{ нулей}}\ \underbrace{1...1}_{(n-k) \text{ единиц}}),\\ 0 \text{ в других случаях;}\end{cases}$$

$$\varphi^n_{\tilde{\sigma}}(x_1,...,x_n)=\begin{cases}2 \text{ при } \tilde{x}=\tilde{\sigma},\\ 0 \text{ в других случаях,}\end{cases}\qquad(2)$$

функции $\varphi^n_{\tilde{\sigma}}(\tilde{x})$ при $\tilde{\sigma}\neq\tilde{0},\tilde{\sigma}\neq\tilde{1}$ получаются из $\varphi^n_k(\tilde{x})$ подходящим переименованием переменных; $\varphi^n_{\tilde{0}}(\tilde{x})=\varphi^n_{\vee}(\tilde{x}),\varphi^n_{\tilde{1}}(\tilde{x})=\varphi^n_{\&}(\tilde{x})$.

Пусть $f'(\tilde{x})$ получается из функции $f(\tilde{x})\in\tilde{T}_0$ заменой всех значений функции, равных 2, на 0. Тогда $f'(\tilde{x})$ реализуется в базисе $\{x\oplus y,xy\}$.

Отсюда для произвольной функции $f(\tilde{x})$ из $\tilde{T}_0$ имеем:

$$f(x_1,...,x_n)=f'(\tilde{x})\vee\bigvee_{\tilde{\sigma}:f(\tilde{\sigma})=2}\varphi^n_{\tilde{\sigma}}(\tilde{x}).$$

Этим доказано следующее утверждение.

Утверждение. Класс $\tilde{T}_0$ порождается конечной системой 2-го порядка $\{x\oplus y,\ xy,\ \varphi_{\vee}(x,y),\ \varphi_{\&}(x,y),\ \varphi_{01}(x,y)\}$.

Следствие. Класс $\tilde{T}_1$ всех функций, сохраняющих предикат $x\neq 0$, имеет порядок 2.

Порядок класса $\tilde{T}_{01}$. Класс $\tilde{T}_{01}$ всех функций, сохраняющих предикат $((x,y)=(0,1))\vee(x=2)\vee(y=2)$, является расширением класса T_{01} (обозначение С.В. Яблонского) всех булевых α-функций.

При отождествлении всех переменных функции из $\tilde{T}_{01}$ получается одна из функций одного переменного $\binom{0\,1}{0\,1},\binom{0\,1}{0\,2},\binom{0\,1}{2\,0},\binom{0\,1}{1\,2},\binom{0\,1}{2\,1},\binom{0\,1}{2\,2}$.

Пусть мы имеем $f(x,...,x)=x, f(\tilde{x})\in\tilde{T}_{01}$.

Определим функции:

$$f_1(x_1,...,x_n)=\begin{cases}1, \text{ если } f(x_1,...,x_n)=2,\\ f(x_1,...,x_n) \text{ в других случаях;}\end{cases}$$

$$f_2(x_1,...,x_n)=\begin{cases}0, \text{ если } f(x_1,...,x_n)=2,\\ f(x_1,...,x_n) \text{ в других случаях.}\end{cases}$$

По построению $f_1(\tilde{x})$ и $f_2(\tilde{x})$ принадлежат T_{01} и порождаются системой $\{x\oplus y\oplus z,xy\}$. Тогда $f(\tilde{x})=\varphi'(f_1(\tilde{x}),f_2(\tilde{x}))$,

где $\varphi'(x,y)=\begin{cases}2 \text{ при } x\neq y,\\ x \text{ при } x=y.\end{cases}$

$\varphi'(x,y)$ также принадлежит $\tilde{T}_{01}$. Таким образом, в этом случае $f(\tilde{x})$ порождается системой $\{x\oplus y\oplus z,\ xy,\ \varphi'(x,y)\}$.

Если $f(x,...,x)\neq x$, то верно $f(\tilde{x})\in\tilde{T}_0$ или $f(\tilde{x})\in\tilde{T}_1$, поэтому $f(\tilde{x})$ порождается системой $\{x\oplus y,\ xy,\ \varphi_{\vee}(x,y),\ \varphi_{\&}(x,y),\ \varphi_{01}(x,y)\}$ либо системой $\{x\oplus y\oplus 1,\ x\vee y,\ \varphi^*_{\vee}(x,y),\ \varphi^*_{\&}(x,y),\ \varphi^*_{01}(x,y)\}$ соответственно. В данном случае $\varphi^*(x,y)$ обозначает двойственную функцию к функции $\varphi(x,y)$ [3].

Поскольку класс T_{01} имеет порядок 3 [3], а функции из T_{01} выражаются только над базисами из T_{01}, то нет базисов класса $\tilde{T}_{01}$ порядка меньше 3. Отсюда класс $\tilde{T}_{01}$ имеет порядок 3.

Порядок класса Фрейвалда Φ_1. Класс Фрейвалда Φ_1 – все булевы всюду определенные функции и функции нигде не определенные порождаются системой $\{\overline{xy},2\}$ второго порядка.

Порядок класса $\tilde{M}$. Класс $\tilde{M}$ – класс всех частичных функций, которые могут быть доопределены до монотонных. Покажем, что класс $\tilde{M}$ порождается системой 2-го порядка $\{xy,\ x\vee y,\ 0,1,\ \varphi_{\vee}(x,y),\ \varphi_{\&}(x,y),\ \varphi_{01}(x,y)\}$.

Пусть $f(\tilde{x})$ – произвольная функция из $\tilde{M}$, $F(\tilde{x})$ – соответствующая ей монотонная булева функция, $F(\tilde{x})$ порождается системой $\{xy,\ x\vee y,\ 0,1\}$. Система функций $\{xy,\ x\vee y,\ \varphi_{\vee}(x,y),\ \varphi_{\&}(x,y),\ \varphi_{01}(x,y)\}$ порождает функции (2).

Тогда имеем $f(\tilde{x})=F(\tilde{x})\vee\bigvee_{\tilde{\sigma}:f(\tilde{\sigma})=2}\varphi^n_{\tilde{\sigma}}(\tilde{x})$, что и требовалось показать.

Порядок класса $\widetilde{S}$**.** Класс $\widetilde{S}$ – класс всех частичных функций, которые могут быть доопределены до самодвойственных.

Пусть $f(\widetilde{x}) \in \widetilde{S}$ и $F(\widetilde{x})$ – доопределение $f(\widetilde{x})$ до самодвойственной функции, $F(\widetilde{x}) \in D_3$, D_3 – замкнутый класс всех самодвойственных функций [3].

Введем функции:

$$G^{\widetilde{\alpha}}(\widetilde{x}) = \begin{cases} F(\widetilde{x}) \text{ при } \widetilde{x} = \widetilde{\alpha}, \widetilde{x} = (\overline{\alpha}_1,...,\overline{\alpha}_n), \\ \overline{F}(\widetilde{x}) \text{ в других случаях;} \end{cases}$$

$$f_0(x,y,z) = \begin{cases} 2 \text{ при } x = y = z = 0, \\ x \oplus y \oplus z \text{ в других случаях,} \end{cases}$$

$$f_1(x,y,z) = \begin{cases} 2 \text{ при } x = y = z = 1, \\ x \oplus y \oplus z \text{ в других случаях,} \end{cases}$$

$$f_2(x,y,z) = \begin{cases} 2 \text{ при } x = y = z, \\ x \oplus y \oplus z \text{ в других случаях.} \end{cases}$$

Самодвойственные функции $F(\widetilde{x})$ и $G^{\widetilde{\alpha}}(\widetilde{x})$ реализуемы в базисе $\{\overline{x}, h_2(x,y,z)\}$, где $h_2(x,y,z) = xy \vee yz \vee zx$ – мажоритарная функция. Кроме того, можно выразить функции:

$$f_1(x,y,z) = \overline{f_0(\overline{x},\overline{y},\overline{z})},$$

$$f_2(x,y,z) = h_2(f_0(x,y,z), f_0(x,y,z), f_1(x,y,z)).$$

Рассмотрим функции из $\widetilde{S}$:

$$H_0^{\widetilde{\alpha}}(\widetilde{x}) = f_0\,(F(\widetilde{x}), G^{\widetilde{\alpha}}(\widetilde{x}), G^{\widetilde{\alpha}}(\widetilde{x})),$$

$$H_1^{\widetilde{\alpha}}(\widetilde{x}) = f_1(F(\widetilde{x}), G^{\widetilde{\alpha}}(\widetilde{x}), G^{\widetilde{\alpha}}(\widetilde{x})),$$

$$H_2^{\widetilde{\alpha}}(\widetilde{x}) = f_2\,(F(\widetilde{x}), G^{\widetilde{\alpha}}(\widetilde{x}), G^{\widetilde{\alpha}}(\widetilde{x})).$$

Таким образом, имеем функции, которые могут быть доопределены до самодвойственной функции $F(\widetilde{x})$ путем замены единственного значения 2 на 0 $\{H_0^{\widetilde{\alpha}}(\widetilde{x})\}$, 2 на 1 $\{H_1^{\widetilde{\alpha}}(\widetilde{x})\}$, двух 2 на противоположных наборах на 0 и 1 $\{H_2^{\widetilde{\alpha}}(\widetilde{x})\}$.

Пусть $\widetilde{\alpha}^1,...,\widetilde{\alpha}^r$ – все наборы, в которых $f(\widetilde{x})$ не определена.

Разобьем их на три группы:

1) группа N_0, где $f(\overline{\alpha}_1,...,\overline{\alpha}_n) = 1$,

2) группа N_1, где $f(\overline{\alpha}_1,...,\overline{\alpha}_n) = 0$,

3) группа N_2, где $f(\overline{\alpha}_1,...,\overline{\alpha}_n) = 2$.

Для наборов из N_0 имеем:

$$H_0^{\widetilde{\alpha}}(\widetilde{x}) = \begin{cases} 2 \text{ при } \widetilde{x} = \widetilde{\alpha}, \\ F(\widetilde{x}) \text{ в других случаях,} \end{cases}$$

для наборов из N_1 имеем:

$$H_1^{\widetilde{\alpha}}(\widetilde{x}) = \begin{cases} 2 \text{ при } \widetilde{x} = \widetilde{\alpha}, \\ F(\widetilde{x}) \text{ в других случаях,} \end{cases}$$

для наборов из N_2 имеем:

$$H_2^{\widetilde{\alpha}}(\widetilde{x}) = \begin{cases} 2 \text{ при } \widetilde{x} = \widetilde{\alpha}, \widetilde{x} = (\overline{\alpha}_1,...,\overline{\alpha}_n), \\ F(\widetilde{x}) \text{ в других случаях.} \end{cases}$$

Пусть функция $f(\widetilde{x})$ имеет нечетное число значений 2. Тогда

$$f(\widetilde{x}) = \sum_{\widetilde{\alpha} \in N_0} H_0^{\widetilde{\alpha}}(\widetilde{x}) \oplus \sum_{\widetilde{\alpha} \in N_1} H_1^{\widetilde{\alpha}}(\widetilde{x}) \oplus \sum_{\widetilde{\alpha} \in N_2} H_2^{\widetilde{\alpha}}(\widetilde{x}).$$

Если $f(\widetilde{x})$ имеет четное число значений 2, то, переобозначив функции $H_i^{\widetilde{\alpha}}(\widetilde{x}), i = \overline{1,3}$, составим перечень $\psi_1(\widetilde{x}),...,\psi_{2k}(\widetilde{x})$. Отсюда

$$f(\widetilde{x}) = \sum_{i=1}^{2k-2} \psi_i(\widetilde{x}) \oplus h_2(\psi_{2k-1}(\widetilde{x}), \psi_{2k-1}(\widetilde{x}), \psi_{2k}(\widetilde{x})).$$

Таким образом, класс $\widetilde{S}$ порождается системой $\{\overline{x},\ h_2(x,y,z),\ f_0(x,y,z)\}$.

Поскольку класс S имеет порядок 3 [3], а функции из S выражаются только над базисами из S, то нет базисов класса $\widetilde{S}$ порядка меньше 3. Отсюда класс $\widetilde{S}$ имеет порядок 3.

Класс $\widetilde{O}_9$**.** Класс $\widetilde{O}_9$ – класс всех функций, сохраняющих множество (основание) функций 2-х переменных $\{0, 1, x, y, \overline{x}, \overline{y}, g(x,y)\}$, где $g(x,y)$ – множество всех функций, неопределенных хотя бы на одном наборе.

Покажем, что класс $\widetilde{O}_9$ не имеет конечного базиса. Предположим противное. Пусть $\exists\, k \geq 2$ такое, что множество всех функций от не более k переменных $B = \{\varphi_m(x_1,...,x_k)\}$, где m – номер функции, является базисом в $\widetilde{O}_9$. Классу $\widetilde{O}_9$ принадлежат функции последовательности

$$g_n(x_1,...,x_n) = \begin{cases} x_n \text{ при } \widetilde{x} \neq \widetilde{1}, \\ 2 \text{ при } \widetilde{x} = \widetilde{1}, \end{cases} \qquad (3)$$

где $n = k+1, k+2, \ldots$.

Тогда функцию $g_{k+1}(x_1,...,x_{k+1})$ можно выразить с помощью некоторой суперпозиции над базисом B, при построении которой используется функция $\varphi_i(\widetilde{x}^i)$, зависящая существенно от l переменных, причем $2 \leq l \leq k$. Поскольку $\varphi_i(\widetilde{x}^i) \in \widetilde{O}_9$, существует набор $\widetilde{\tau}^i$ такой, что $\varphi_i(\widetilde{\tau}^i) = 2$. Кроме того, имеется переменная x_j, от которой существенно не зависит $\varphi_i(\widetilde{x}^i)$, но зависит $g_{k+1}(x_1,...,x_{k+1})$. Значит, рассматриваемая суперпозиция не определена хотя бы на двух наборах: $\widetilde{\tau}_1$ (включающем $\widetilde{\tau}^i$ и $x_j = 0$) и $\widetilde{\tau}_2$ (включающем $\widetilde{\tau}^i$ и $x_j = 1$). Это противоречит (3).

Следовательно, B – не базис, вопреки предположению.

Класс $\widetilde{L}_1$. Класс $\widetilde{L}_1$ – класс всех частичных функций, которые могут быть доопределены до линейных булевых функций. Докажем, что $\widetilde{L}_1$ не имеет конечного базиса. Предположим противное: $B=\{\varphi_m(x_1,...,x_k)\}$, m – номер функции, $k\geq 2$ – базис, состоящий из всех частичных функций из $\widetilde{L}_1$, зависящих от не более k переменных.

Пусть $\widetilde{\psi}(x_1,...,x_n)$ – линейный оператор из k булевых линейных функций. Число решений системы линейных булевых уравнений

$$\widetilde{\psi}(x_1,...,x_n)=(\sigma_1,...,\sigma_k) \qquad (4)$$

равно четному числу или единице (либо 0 решений, если система (4) несовместна, либо 2^{n-r} решений, если r – ранг системы).

Если $\varphi_i(x_1,...,x_k)\in B$ и $\exists(\sigma_1,...,\sigma_k)$ такое, что $\varphi_i(\sigma_1,...,\sigma_k)=2$, то число решений уравнения

$$\varphi_i(\widetilde{\psi}(x_1,...,x_n))=2 \qquad (5)$$

равно числу наборов, в которых $\varphi_i(x_1,...,x_k)$ не определено, помноженному на 2^{n-r} [при предположении существования решения системы (4)].

Замечание. Нам в дальнейшем неважно оценить точно число решений уравнения (5). Важно отметить, что это число четное (или единица) и неравное нулю, хотя априори оно может равняться нулю, но тогда $\varphi_i(x_1,...,x_k)$ можно будет заменить линейной всюду определенной функцией.

Пусть $f(\widetilde{x})$ – произвольная функция из $\widetilde{L}_1$. Ее можно представить суперпозицией вида

$$f(\widetilde{x})=F(\varphi_i(\widetilde{\psi}(x_1,...,x_n)),x_1,...,x_n),$$

где для $\varphi_i(x_1,...,x_k)$ $\exists\widetilde{\sigma}$ такое, что $\varphi_i(\sigma_1,...,\sigma_k)=2$ и система $\widetilde{\psi}(x_1,...,x_n)=\widetilde{\sigma}$ имеет решение (следует из замечания выше). Следовательно, $f(\widetilde{x})$ не является функцией вида (3) с одной двойкой, принадлежащей классу $\widetilde{L}_1$, ни в случае, когда число решений уравнения (5) четное, ни в случае, когда решение единственное (тогда это можно показать аналогично рассуждениям для класса $\widetilde{O}_9$). Таким образом, множество B не полное в $\widetilde{L}_1$, поэтому не является базисом, и, значит, класс $\widetilde{L}_1$ не имеет конечного базиса, что и требовалось доказать.

Заключение. В настоящей работе исследованы порядки всех предполных классов (8 классов) алгебры $\widetilde{P}_2$ – частичных булевых функций; построены базисы указанного порядка для шести классов и указаны эффективные процедуры для выразимости функций класса над базисом. В двух случаях оказалось, что конечного базиса не существует и выразимость функций этого класса возможна лишь для конечного алфавита переменных.

Библиографический список

1. *Post E.L.* Introduction to a general theory of elementary propositions // Amer. J. Math. – 1921. – V. 43, № 4. – P.163-185.

2. *Post E.L.* Two-valued iterative systems of mathematical logic // Annals of Math. Studies. – Princeton Univ. Press, 1941. – V. 5.

3. *Яблонский С.В., Гаврилов Г.П., Кудрявцев В.Б.* Функции алгебры логики и классы Поста. – М.: Наука, 1966. – 120 с.

4. *Яблонский С.В.* Функциональные построения в k-значной логике // Труды математического института им. В.А.Стеклова. – М.: Наука, 1958. – Т. 51. – С. 5-142.

5. *Гниденко В.Н.* Нахождение порядков предполных классов в трехзначной логике // Проблемы кибернетики. – Вып. 8. – М.: Наука, 1962. – С. 341-346.

6. Lau D. Submaximalle Klassen Von P_3 // Electron. Informationsverarb. und Kybern. – 1982. – Вып. 18. – № 4-5. – S. 227-243.

7. *Михеева Е.А.* Построение в P_k максимальных классов, не имеющих базисов // Дискретная математика. – 1998. – Т.10. – Вып.2. – С.137-159.

8. *Саблина В.А.* О порядках замкнутых классов в трехзначной логике, I // Вестник Рязанского государственного радиотехнического университета № 2 (Вып. 24). – Рязань, 2008. – С. 121-124.

9. *Саблина В.А.* О порядках замкнутых классов в трехзначной логике, II // Вестник Рязанского государственного радиотехнического университета № 4 (Вып. 26). – Рязань, 2008. – С. 84-87.

10. *Мальцев А.И.* Итеративная алгебра и многообразия Поста // Алгебра и логика. – Новосибирск: Наука, 1966. – Т. 5. – № 2. – С. 5-24.

11. *Фрейвалд Р.В.* Функциональная полнота для не всюду определенных функций алгебры логики // Дискретный анализ. – Новосибирск: Наука, 1966. – Вып. 8. – С. 55-68.

УДК 004.9

Д.В. Суворов, С.А. Батуркин

АРХИТЕКТУРА СИСТЕМ УДАЛЕННОГО ДОСТУПА К УЧЕБНОМУ ЛАБОРАТОРНОМУ ОБОРУДОВАНИЮ НА БАЗЕ СТАНДАРТА SCORM

Рассмотрен стандарт обмена учебным контентом SCORM, приведены примеры основных типов существующих архитектур систем удаленного доступа к учебному лабораторному оборудованию различного типа, использующие стандарт SCORM, предложена схема типового решения архитектуры для создания удаленных учебных лабораторий на базе стандарта.

***Ключевые слова**: удаленный доступ, учебный, SCORM, архитектура, масштабируемый, дистанционный, лабораторный.*

Введение. Основным инструментом взаимодействия ученых в рамках крупных проектов все чаще становятся системы удаленного доступа. Развитие информационных технологий позволяет создавать все более и более совершенные инструменты научных исследований.

Прошло более 20 лет с момента появления первых решений в области систем удаленного доступа к лабораторному оборудованию и в настоящее время появилась возможность выделения основных типов архитектур.

Развитие систем удаленного доступа потребовало решения задачи масштабирования в рамках отдельных проектов и направления в целом. Это в свою очередь породило потребность в стандартизации и унификации методологий удаленного доступа к учебному лабораторному оборудованию, разработки типовых решений и инструкций на этой основе.

Целью данной работы является выделение типовых архитектурных решений, разработка масштабируемой методологии удаленного доступа с использованием единой базы для стандартизации и унификации обмена данными. Для решения указанных задач необходимо рассмотрение нормативной базы в области обмена учебным контентом и построения обучающих систем, а также мирового опыта построения систем удаленного доступа к лабораторному оборудованию.

Стандарт SCORM (Sharable Content Object Reference Model). Стандарт разрабатывается инициативной группой ADL (Advanced Distributed Learning) с 1997 года при поддержке министерства обороны США [1]. Данный стандарт считается одной из самых современных версий стандартов обмена учебными материалами.

Стандарт SCORM неразрывно связан с понятием системы управления обучением (LMS – learning management system). Термин “LMS”, используемый в SCORM, обозначает набор функциональных возможностей, разработанных для распространения, контроля и управления образовательным контентом и учебным процессом. Этот термин относится к простым и сложным системам управления.

ADL лаборатории [2] проверяют все свои продукты на предмет доступности, интероперабельности, возможности использоваться повторно, длительно и эффективно. Эти критерии касаются следующих особенностей.

1. Способность перемещать образовательный Web контент в любую среду вне зависимости от прикладной программы.

2. Многократно использовать контент в любой среде вне зависимости от прикладной программы.

3. Создание образовательного контента доступного и легко поддающегося поиску вне зависимости от прикладных программ.

4. Применение SCORM к образовательным программам.

По своей сути SCORM – это собрание спецификаций и стандартов, которые были собраны в несколько “технических книг”, как показано на рисунке 1.

В контексте SCORM широко используются LMS приложения. SCORM сосредотачивается на интерфейсе содержания и LMS среде, но не касается особенностей LMS. В SCORM термин LMS подразумевает среду сервера. Другими словами, в SCORM LMS определяет какую информацию и куда поставить и отслеживает работу пользователя с материалом.

Образовательный контент в SCORM понимается как небольшие образовательные объекты, собранные в курсы, главы, модули, задания и т.п. Эти единицы содержания (Content Objetcs), собранные из более мелких образовательных объектов, разработаны таким образом, что могут быть использованы многократно в разных контекстах.

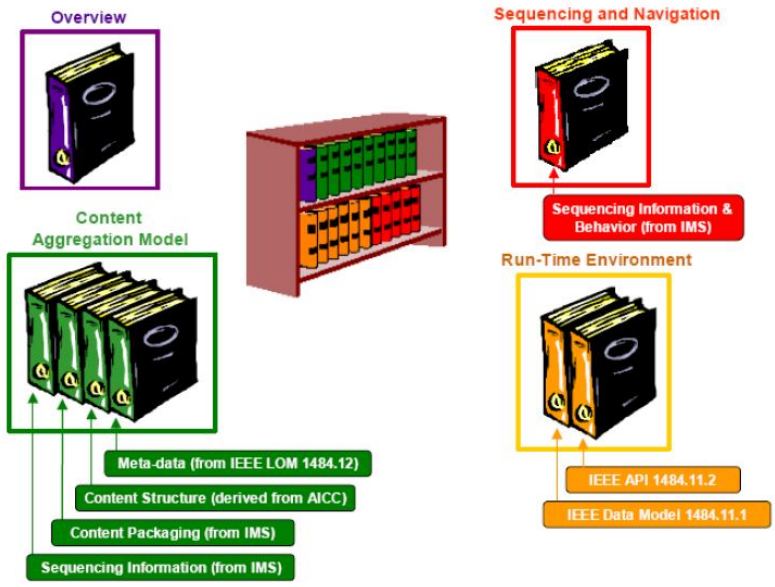


Рисунок 1 – Структура SCORM

Такой подход означает, что навигация и упорядочение единиц содержания происходит по правилам, определенным при накоплении материала. LMS работает согласно внешне определенным правилам организационной структуры, не имея данных об организации самого документа. Это позволяет разработчику определять правила упорядочения и навигации, сохраняя возможность использовать образовательный ресурс многократно и в различных конфигурациях и контекстах. Разделив правила навигации и упорядочения единиц содержания, SCORM дает возможность использовать материал в различных учебных целях.

Развитие решений, использующих SCORM, происходит как за рубежом, так и в России, потому рассмотрим два достаточно развитых проекта.

Типовые архитектурные решения при создании лабораторий удаленного доступа, использующие стандарт SCORM. Первым успешным экспериментом по созданию учебных лабораторий удаленного доступа можно считать проект iLabs Массачусетского технологического института [3].

Целью архитектурного решения является поддержка пакетных экспериментов, которые можно распространить за пределы лаборатории.

Архитектура представляет собой трехуровневое веб-решение, представленное на рисунке 2.

1. Первый уровень - уровень клиентского приложения студента (клиента), которое работает как апплет или загруженное приложение на рабочей станции студента (клиента).

2. Средний уровень, называемый Service Broker, как правило, находится на сервере на территории кампуса студента (клиента). Он поддерживает стандартные реляционные базы данных, такие как SQL Server или Oracle. Клиент студента общается исключительно с Service Broker, который направляет спецификации эксперимента к началу третьего уровня.

3. Третий уровень - сервер самой лаборатории, который выполняет указанные эксперименты и уведомляет Service Broker, когда результаты будут готовы для их загрузки.

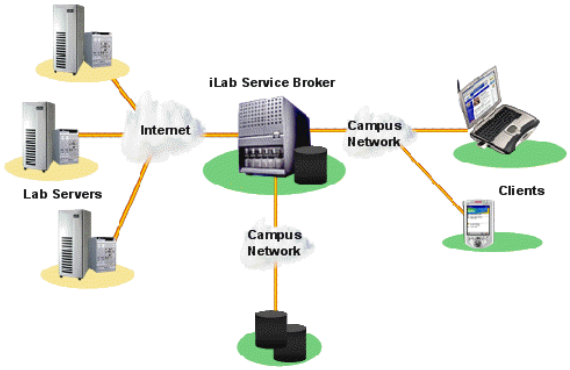


Рисунок 2 – Архитектура iLabs

В этой схеме клиент клиента и серверы лаборатории представляют область или лабораторно-зависимые программные модули. Service Broker написан с использованием полностью универсального кода. Интерфейс реализуется через веб-службы SOAP, через каналы в WSDL. Клиент клиента обновляет модули или лабораторно - зависимые части кода, которые должны понимать протокол для определения параметров конкретного эксперимента.

Сервер лаборатории не хранит персональных данных клиентов, использующих систему, а

временно хранит только спецификации эксперимента и результаты.

Для установления надежного канала связи между сервером лаборатории и Service Broker, в первую очередь, выполняется проверка подлинности, и Service Broker удостоверяет сертификат клиента.

Структура iLabs, благодаря использованию SCORM, имеет модульную масштабируемую структуру для пакетного эксперимента, в которой станция клиента никогда не связывается с сервером лаборатории непосредственно. Модульная структура iLabs изображена на рисунке 3.

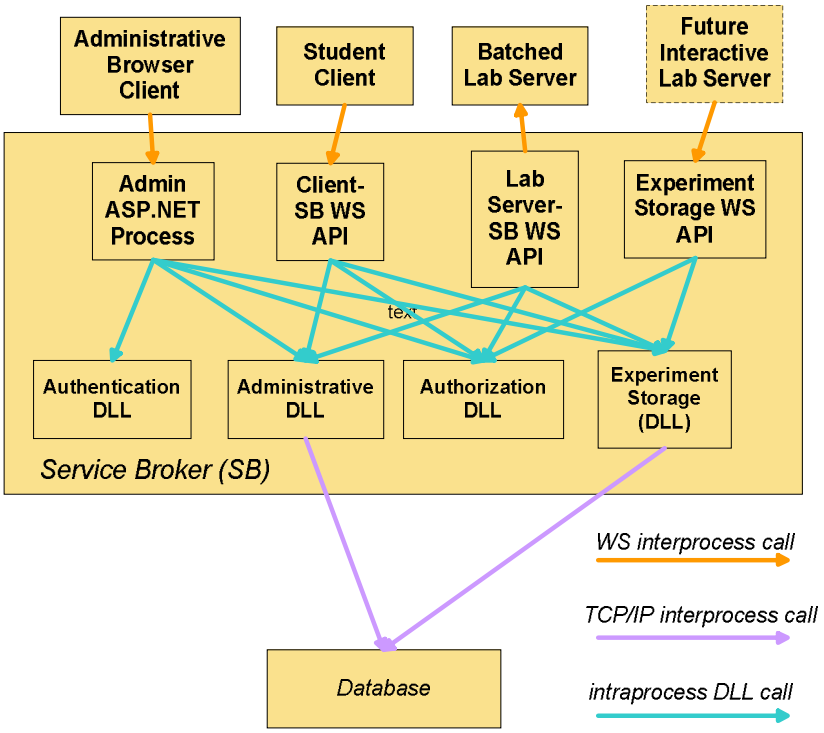


Рисунок 3 – Модульная масштабируемая структура iLabs

Это возможно благодаря тому, что пакетный эксперимент требует небольшого количества транзакций между клиентом и сервером лаборатории. Концептуально выполнение эксперимента требует одну двустороннюю транзакцию, хотя фактические протоколы веб-службы являются более сложными. Эта простота не распространяется на случаи потокового датчика эксперимента или для интерактивного эксперимента, для которых существуют веские аргументы для того, чтобы поддерживать постоянное подключение к серверу лаборатории непосредственно.

Стоит отметить тот факт, что архитектура iLabs, благодаря набору стандартных модулей и использованию максимального количества программных модулей, написанных с применением наиболее универсальных кодов, является одним из наиболее интересных кандидатов в масштабируемые системы.

Еще одним важным преимуществом подобной архитектуры и организации лаборатории является исполнение пакетных экспериментов, что позволяет существенно снизить нагрузку на серверные элементы.

Однако, как уже было сказано, отсутствие режима потокового эксперимента не позволяет полностью использовать все возможности данного типа архитектуры, даже при использовании SCORM.

Указанного выше недостатка лишены лаборатории удаленного доступа МГТУ им. Н.Э. Баумана [4].

В составе сети удаленных лабораторий университета находятся:

1) лаборатория НОЦ «Нанотехнологические системы и наноэлектроника»;
2) лаборатория спектрометрии;
3) лаборатория испытаний материалов;
4) лаборатория удаленного управления радиотелескопом;
5) интернет - лаборатория"Робототехника".

В алгоритме сервисных программ можно выделить два больших блока. Первый из них поддерживает работу Web-сервера практикума, а второй - обеспечивает его связь со стендом, выполнение эксперимента по сценариям удаленных пользователей и трансляцию полученных результатов.

Прикладное программное обеспечение включает в себя ряд подсистем, функционально обслуживающих разные составляющие практи-

кума. Все эти подсистемы объединены в интерактивную диалоговую удаленную систему (ИНДУС), архитектура которой показана на рисунке 4, разработанную в МГТУ им. Н.Э.Баумана.

ИНДУС реализует ключевые моменты стандарта SCORM в процессе обмена данными с заказчиком эксперимента, как правило, – учащимся или группой учащихся.

Рисунок 4 – Состав системы ИНДУС

1. Подсистема телекоммуникаций обеспечивает связь удаленного пользователя с Web-сервером и Web-сервера с управляющим компьютером.

2. Обучающая подсистема содержит полную информацию об экспериментальном стенде, краткие теоретические положения, методику измерения и т.д. в объеме, достаточном для подготовки к проведению лабораторной работы и написания отчета. Использование спецификаций SCORM позволило создать единый формат вывода данных и унифицированную базу данных об экспериментах и имеющемся оборудовании.

3. Подсистема тестирования предназначена для контроля усвоения знаний о стенде, физических принципах и методике эксперимента, без которого студент не допускается к активному проведению опытов. В данной системе, наряду со SCORM, активно используется спецификация IMS QTI, позволяющая создавать стандартизованные тестовые задания.

4. Справочная подсистема содержит текстовые, табличные и графические данные, необходимые для обработки результатов

5. Подсистема идентификации пользователя проверяет, имеет ли пользователь право на управление установкой в настоящий момент, и обеспечивает проведение эксперимента в данное время только одним пользователем.

6. Подсистема программирования условий эксперимента позволяет в интерактивном режиме настроить стенд на требуемые условия проведения опытов.

7. Подсистема имитации эксперимента позволяет до проведения активных экспериментов знакомиться с пультом управления стендом и имитировать элементарные операции настройки условий эксперимента, чтобы снизить затраты времени на реальный эксперимент.

8. Подсистема визуализации данных эксперимента позволяет наглядно представить результаты эксперимента в форме, удобной для их дальнейшей обработки.

9. Подсистемы управления и измерения позволяют перенастраивать лабораторный стенд и осуществлять его функционирование в заданном пользователем режиме работы, а также осуществлять измерение заданных параметров.

10. Объектовая подсистема и подсистема измерений представляют собой стендовую часть лабораторной установки.

Почти все указанные модули так или иначе используют SCORM в организации обмена

учебным контентом.

Приведенная архитектура, так же как и аналогичная iLab, позволяет в значительной степени производить масштабирование экспериментов, притом пределы масштабирования могут быть ограничены только техническими характеристиками оборудования и каналов связи.

Типовая структура лаборатории удаленного доступа к учебному лабораторному оборудованию на базе SCORM. Стандарт получил широкое распространение и активно используется в следующих системах обучения [5]:

1) WebTutor (плафформы Lotus и Microsoft);
2) Adobe Connect Pro;
3) Oracle Enterprise Learning Management System;
4) IBM Workplace Collaborative Learning;
5) SAP Learning Solution;
6) Moodle;
и пр.

Ряд вузов и научных центров имеют свои разработки в области дистанционного управления учебным оборудованием, основанные на стандарте SCORM. Все эти решения позволяют выделить типовую архитектуру системы удаленного доступа к учебному лабораторному оборудованию на базе SCORM [1] .

Типовая архитектура системы удаленного доступа представлена на рисунке 5.

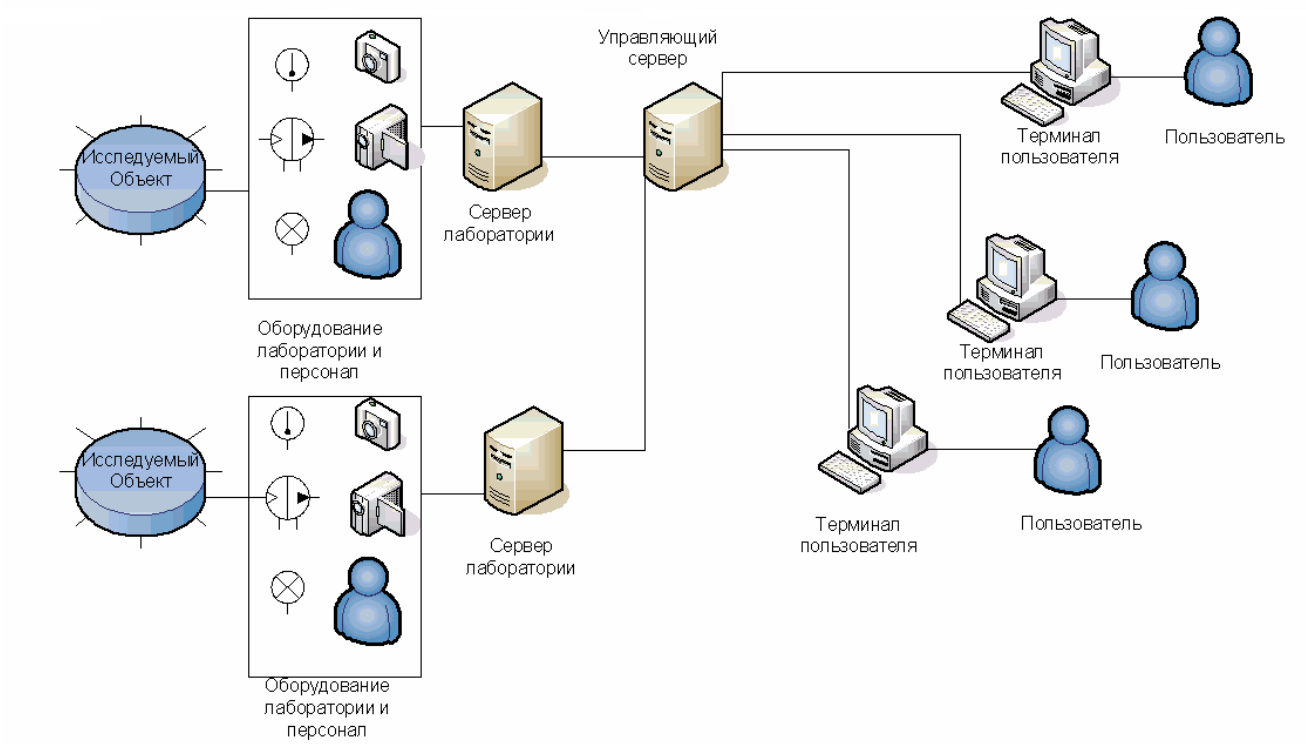


Рисунок 5 – Архитектура типового решения для организации удаленного доступа к научным лабораториям

1. Лабораторная установка. Оборудование, на котором выполняется соответствующий эксперимент.

2. Web - сервер. Обеспечивает доступ к основному объему методического материала по соответствующей теме в сети Internet. Осуществляет маршрутизацию движения удаленного исполнителя по необходимым серверам в процессе его работы.

3. SQL сервер. Обеспечивает доступ к базам экспериментальных данных и базам справочных данных, необходимых для выполнения лабораторных и расчетных работ.

4. Лабораторный сервер. Компьютер, обеспечивающий опрос устройства ввода информации и предоставление данных для последующего использования.

5. Устройство ввода/вывода информации. Обеспечивает аналого-цифровые и цифро-аналоговые преобразования сигналов, снимаемых с датчиков лабораторной установки.

6. Блок сопряжения с силовыми сетями и дистанционные пускатели. Служат для обеспечения управляющего воздействия на органы управления.

7. Датчики. Устройства, реагирующие своими чувствительными элементами на изменение исследуемых параметров лабораторной установки и преобразующие данные изменения в удобную для последующей передачи форму.

8. Аудио- и видеоаппаратура. Обеспечивает наглядность эксперимента.

Лабораторная установка оснащена набором датчиков, необходимым для отображения хода протекания эксперимента и обеспечивающим информационную целостность данных с целью их последующей обработки.

Информация со специализированных датчиков через устройства ввода (вывода) поступает на лабораторный сервер.

Установленный на лабораторном сервере виртуальный инструмент, осуществляющий информационный обмен между сервером и клиентом посредством специального интерфейса, производит дистанционный запуск виртуальных инструментов и др.

SQL – сервер – один из важнейших элементов при организации и проведении удаленного эксперимента, потому что благодаря ему идет процесс оперативной обработки данных, хранящихся в базах. Доступ к базам справочных данных является второй важной функцией SQL – сервера.

Блок сопряжения с силовыми сетями и дистанционные пускатели выбираются или конструируются исходя из условий проведения эксперимента и обеспечения надежности и безопасности работы.

Аудио- и видеоаппаратура может быть как общего назначения, так и специализированной, как например видеоаппаратура для съёмки быстротекущих процессов или аудиоаппаратура для съема шумов или аудиосигналов в различных диапазонах.

Программное обеспечение, используемое на сервере лаборатории и управляющем сервере, основано на применении принципов управления контентом, заложенных в SCORM. Это позволяет любому пользователю, имеющему доступ к клиентской части эксперимента, выполнять регламентированные действия. Притом в зависимости от технических характеристик канала связи возможно выполнение как пакетного эксперимента (пример - iLab), так и потокового эксперимента (МГТУ им. Н.Э. Баумана).

Заключение. Использование SCORM в организации лабораторий удаленного доступа к учебному оборудованию позволяет создавать масштабируемые системы. Использование технологии масштабирования позволяет сократить временные издержки при постановке и модернизации новых систем удаленного доступа на 30-40 %.

Использование SCORM и совместимых стандартов в области электронного обучения и при организации удаленного доступа в рамках крупных научных проектов позволяет существенно снизить затраты на проведение экспериментов и в процессе подготовки.

При различных типах проводимых экспериментов финансовые затраты заказчика эксперимента снижаются на 10-15 %.

Библиографический список

1. Гуров В.С., Гостин А.М., Батуркин С.А., Суворов Д.В. Международные стандарты и спецификации в дистанционном обучении и исследованиях// Вестник РГРТУ. № 4 (выпуск 34). Рязань, 2010; С. 51-55.

2. http://www.adlnet.org - официальный сайт ADL;

3. http://ilab.mit.edu. – официальный сайт проекта iLabs;

4. http://www.alpud.ru - официальный сайт автоматизированных лабораторных практикумов с удаленным доступом МГТУ им. Н.Э. Баумана;

5. http://dstudy.ru/scorm - портал, посвященный вопросам дистанционного обучения.